PROYECTO SISTEMA ESTADÍSTICO

Manual de instalación y de operaciones

TRANSMILENIO S.A.

Versión 1.0

SEPTIEMBRE DE 2023

Contenido

1. Información general 2
2. Generalidades operacionales del sistema 2
2.1 Operación del sistema 2
3.1Creación de las tablas 3
3.2 Creación del usuario: 3
Creación de los esquemas: 3
3.3 Configuración de Jenkins 4
3.4 Generación del JAR: con el comando mvn clean package 4
3.5 Condiciones para la correcta creación del JAR: 6
3.6 Ejecución del JAR: 6
3.6.1 Ejecución del JAR por consola 7
3.6.2 Ejecución del JAR con el uso de un template.bat 8
4. Servicios web 9
5. Especificaciones del equipo de pruebas 10
5.1 Cómo aumentar la memoria Java en Windows 11
6. Pruebas de ejecución con el comando mvn clean install 11
7. Ejecución en Docker 13
7.1 Configuración del Dockerfile 13
7.2 Ejecución del JAR en Docker 14
8. Proceso de solicitud POST 15
8.1 Lista de los servicios por módulo 16
8.3 Funcionalidad de ejecución por modelo "cron" 17
9. Contenido de las carpetas: 18
9.1 Diagramas: 18
9.2 Documentación: 18
9.3 Excels: 18
9.4 Scripts_borrado: 19
9.5 Scripts_consulta: 19
9.6 Scripts_creación: 19
9.7 application.properties: 20

1. Información general

1.1 introducción del sistema

El sistema estadístico es un sistema que permite procesar las fuentes de información de las diferentes áreas de Transmilenio S.A. generados por los usuarios encargados de llevar el control de la información de la entidad, el sistema realiza un procesamiento con dichos datos y finalmente los almacena en una bodega de datos para permitir la visualización de estos a través de una herramienta para generar reportes (Qlik Sense).

Este manual describe el proceso de instalación para el cargue de información hacia los diferentes módulos, proceso realizado con TDD y Spring Batch.

1.2 Referencias del proyecto

En el anexo **SBD Anexo 1 - Requerimientos Funcionales- 2-AGOSTO-2019.docx**, se detallan los indicadores asociados a los distintos módulos, este documento puede ser propenso a modificaciones.

Dentro del proyecto, se estructuraron una serie de documentos que sirven como lineamiento para la organización y arquitectura del software, dentro de estos, existen actas que indican la infraestructura planteada para el despliegue del aplicativo, estos documentos brindan la posibilidad de entender el porqué de la configuración indicada dentro de este documento, para más información, consultar la sección de anexos en este archivo.
Adicionalmente, el código fuente del proyecto se encuentra almacenado en un repositorio de Gitlab, allí se encuentran los proyectos almacenados en su última versión funcional, el acceso al repositorio también se encuentra dentro de los anexos.

2. Generalidades operacionales del sistema

2.1 Operación del sistema

El sistema estadístico es una plataforma que le permitirá a los usuarios de cada área de Transmilenio consultar la información de toda la entidad a través de una

herramienta de generación de reportes, dichos datos serán obtenidos a través de los archivos cargados por los mismos usuarios o la información que sea consumida por el sistema a través de diferentes fuentes como servicios web o base de datos para finalmente ser visualizado desde la herramienta de Qlik Sense según la necesidad de cada área.

3. Ejecución del sistema

3.1Creación de las tablas

El procedimiento para la creación de las tablas es muy similar en todos los módulos, lo único que cambia es la creación del usuario y de los esquemas, que se crean únicamente en el primer módulo, se deben seguir los siguientes pasos:

Lo primero que se debe realizar es la creación del nuevo usuario "estadístico", para esto se debe de ejecutar el siguiente script:

3.2 Creación del usuario:

- Scripts_creacion/usuario_estadistico.sql

Este script debe ser ejecutado únicamente si el usuario NO existe, de existir el usuario no es necesaria la ejecución del script.

Los siguientes pasos se deben de ejecutar con el nuevo usuario:

Creación de los esquemas:

- Ejecutar el script Scripts_creacion/schemas.sql

Este script debe ser ejecuta únicamente si los esquemas NO existen, de existir los esquemas no es necesaria la ejecución del script.

Para la creación de las tablas de cada módulo existen una serie de scripts que se encuentran en la carpeta Scripts_creacion, una vez que se tengan creados el usuario y los esquemas, se deben de ejecutar los siguientes scripts en el mismo orden indicado:

Creación tablas de stage:

1. Ejecutar el script Scripts_creacion/NOMBRE-MÓDULO_stage.sql

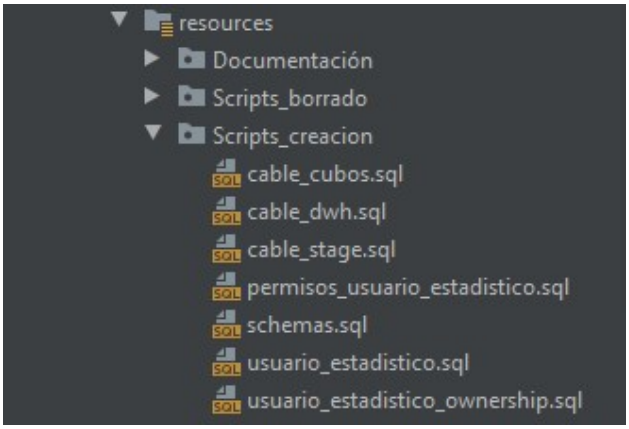
Creación de las tablas de dwh:

2. Ejecutar el script Scripts_creacion/ NOMBRE-MÓDULO_dwh.sql

Creación de las tablas de cubos:

1. Ejecutar el script Scripts_creacion/ NOMBRE-MÓDULO_cubos.sql

Donde NOMBRE-MÓDULO, corresponde al módulo cuyas tablas van a ser creadas. A continuación, se pueden ver los scripts de creación del módulo de cable, este proceso es muy similar en los otros módulos y dentro de los proyectos se cuenta con la misma estructura y jerarquía de directorios:



3.3 Configuración de Jenkins

Para iniciar la configuración de Jenkins se tiene como prerrequisito que se encuentre instalado Jenkins.

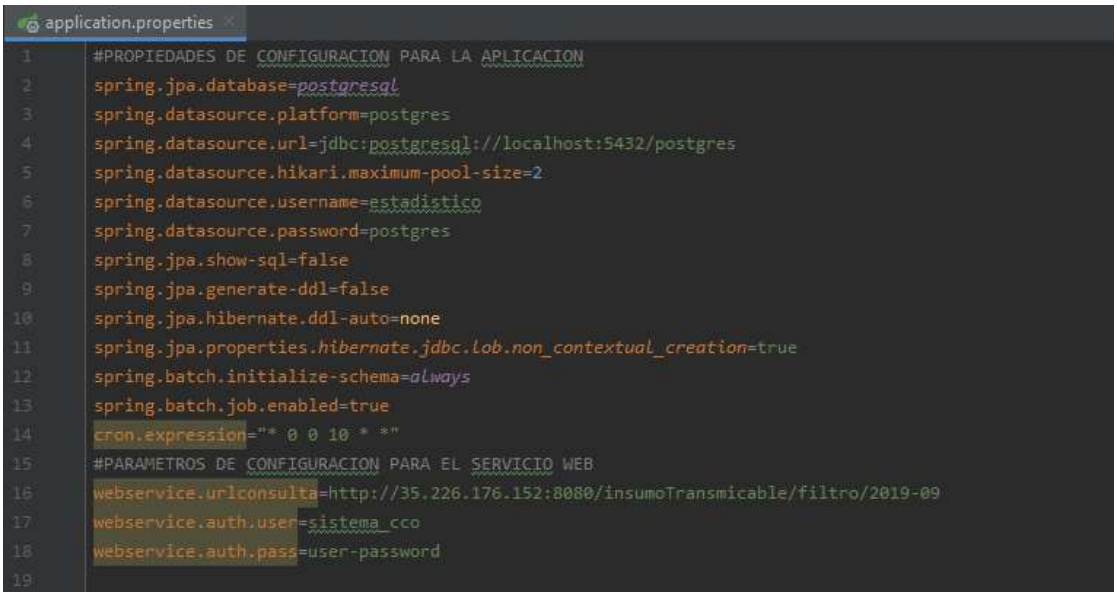
3.4 Generación del JAR: con el comando mvn clean package

Para generar el JAR correspondiente de cada módulo se debe de ejecutar desde la consola el comando: “**mvn clean package**”.

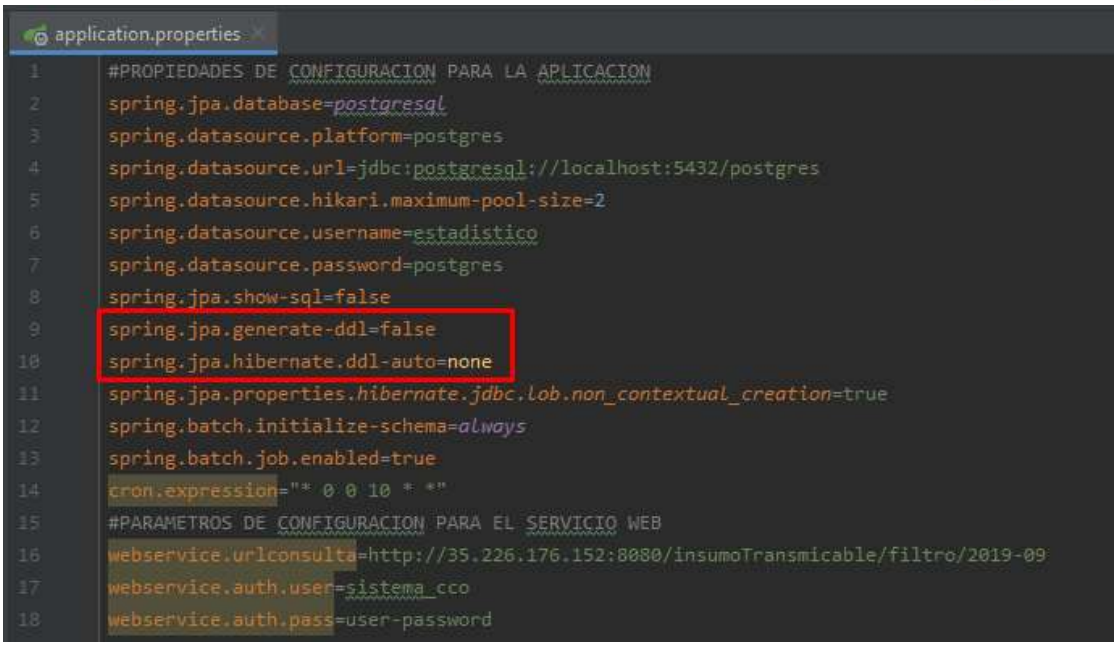
Para la ejecución de este comando se debe configurar previamente lo siguiente:

- Cambiar los valores que se encuentran por defecto en el application.properties de cada módulo por los valores deseados, dentro de este archivo se parametrizan configuraciones asociadas a la conexión de bases de datos, directorios de archivos y url’s de acceso para servicios web.

A continuación, se pueden ver los parámetros que se pueden definir en el módulo de cable, estos van a variar en cada módulo de acuerdo con las necesidades de este:



Las únicas propiedades que no deben de modificarse son las siguientes:



"**spring.jpa.generate-ddl=false**" y "**spring.jpa.hibernate.ddl-auto=none**" que deben permanecer en ***FALSE*** y ***NONE*** respectivamente, porque de lo contrario se borraran elementos necesarios para el cargue y aparecerán errores en la ejecución. Esto aplica para todos los módulos.

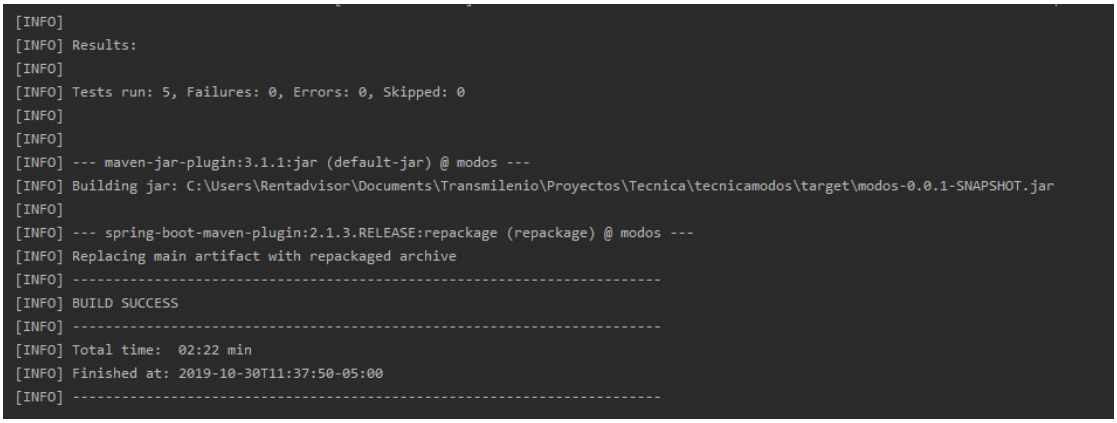
3.5 Condiciones para la correcta creación del JAR:

Para que se genere correctamente el JAR, se deben cumplir las siguientes condiciones:

1. Deben estar creadas las tablas en la base de datos
2. Se debe de haber realizado el cargue de los Excel que se encuentra dentro de cada proyecto en la carpeta "/resources/Excels" y/o los servicios web correspondientes a cada módulo, ya que las pruebas se realizan con los registros que se cargan desde estas fuentes.

Si las tablas no se encuentran o no se ha cargado el archivo aparecerán errores. A pesar de esto si al final de la ejecución aparece el mensaje BUILD SUCCESS se creará el JAR, pero lo recomendado es seguir los pasos anteriormente descritos para evitar la aparición de errores o problemas en la consistencia de los datos más adelante.

A continuación, se presenta como se debería una ejecución correcta con el comando (para el módulo de Técnica Modos):



NOTA: En ninguna de las tablas de los diferentes esquemas se deben realizar inserciones manuales, pues afecta el comportamiento normal del aplicativo y todos los cargues se realizan a través del spring batch y la auditoría que el mismo provee.

Para definir y validar casos deben parametrizarse desde las distintas fuentes origen acordadas.

3.6 Ejecución del JAR:

Existen diferentes formas para realizar la ejecución del JAR, pero para todas se debe cumplir el siguiente requisito:

- Tener instalado Java 8 (64-bits)

3.6.1 Ejecución del JAR por consola

Para la ejecución por consola se debe crear una línea de ejecución con la siguiente estructura:

“java -jar” + NOMBRE-MÓDULO-0.0.1-SNAPSHOT.jar + --propiedad_1=valor_propiedad_1 + “ \ “ + --propiedad_2=valor_propiedad_2 + “ \ “ –propiedad_3=valor_propiedad_3 + “ \ “…

Ejemplo (el siguiente ejemplo corresponde al módulo de económica):

```
18  java -jar economica-0.0.1-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
     \ --spring.datasource.username=estadistico \
    --spring.datasource.password=postgres \
    --spring.batch.job.enabled=false \
    --excelpath.economica=C:\Users\Rentadvisor\Documents\Transmilenio\
    Proyectos\Economica\ExcelPruebas_economica\Economica_DDMMAAAA.xlsx
19
```

```
java -jar economica-0.0.1-SNAPSHOT.jar \ --
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres \ --
spring.datasource.username=estadistico \ --spring.datasource.password=postgres
\ --spring.batch.job.enabled=false \ --excelpath.economica=C:
\Documents\Transmilenio\Proyectos\Economica\ExcelPruebas_economica\Economica_D
DMMAAAA.xlsx
```

La línea que se cree no debe estar separada por saltos de línea, y tampoco debe de contener espacios entre el igual de la propiedad y su correspondiente valor pues se asumirá este espacio como parte del valor de esta.

Se deben de cambiar los valores que se encuentran por defecto en las siguientes propiedades, por los valores deseados:

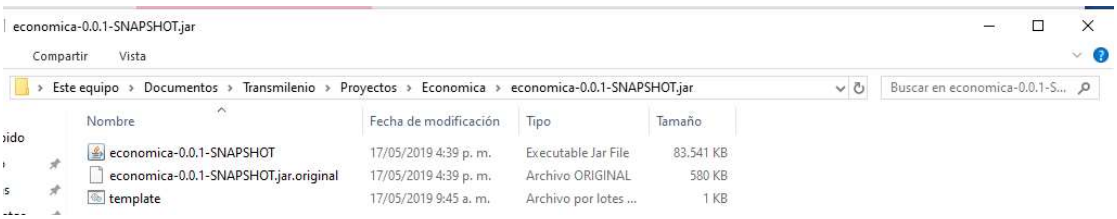
Ejemplo para el caso del módulo de económica:

```
--spring.jpa.database=postgresql
--spring.datasource.platform=postgres
--spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
--spring.datasource.username=estadistico
--spring.datasource.password=postgres
--spring.jpa.show-sql=false
--spring.jpa.generate-ddl=false
--spring.jpa.hibernate.ddl-auto=none
--spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
--
excelpath.economica=C:\\Users\\daav20\\IdeaProjects\\economica\\src\\main\\res
ources\\Economica_DDMMAAAA.xlsx
--excelpath.economica.downloaded = C:\\Users\\RentAdvisor\\Downloads\\
--spring.batch.job.enabled=false
--spring.jpa.open-in-view=false
--webservice.urlstatus=http://35.245.140.104:8002/transmilenio-
api/estadistico/auditoria/{id}
```

Este proceso tiende a variar en los otros módulos.

3.6.2 Ejecución del JAR con el uso de un template.bat

Para la ejecución del JAR con el uso de un fichero con extensión ".bat" se debe de tener en cuenta, que el JAR del módulo que se desea ejecutar y él ".bat" construido deben encontrarse en la misma carpeta:



Para cambiar la configuración de la ruta de los archivos fuente Excel y la conexión a la base de datos, es necesario abrir el archivo template.bat y cambiar los valores que se encuentran por defecto en las siguientes propiedades, por los valores deseados:

Ejemplo para el caso del módulo de económica:

```
--spring.jpa.database=postgresql
--spring.datasource.platform=postgres
--spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
--spring.datasource.username=estadistico
--spring.datasource.password=postgres
--spring.jpa.show-sql=false
--spring.jpa.generate-ddl=false
--spring.jpa.hibernate.ddl-auto=none
--spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
--
excelpath.economica=C:\\Users\\daav20\\IdeaProjects\\economica\\src\\main\\res
ources\\Economica_DDMMAAAA.xlsx
--excelpath.economica.downloaded = C:\\Users\\RentAdvisor\\Downloads\\
--spring.batch.job.enabled=false
--spring.jpa.open-in-view=false
--webservice.urlstatus=http://35.245.140.104:8002/transmilenio-
api/estadistico/auditoria/{id}
```

Este proceso tiende a variar en los otros módulos.

Una vez que se tengan las propiedades con los valores deseados se debe ejecutar template.bat, luego de la ejecución se puede confirmar la correcta ejecución, realizando una consulta en la base de datos.

NOTA: Si en el template.bat se encuentran las propiedades

--spring.jpa.generate-ddl=false

--spring.jpa.hibernate.ddl-auto=none

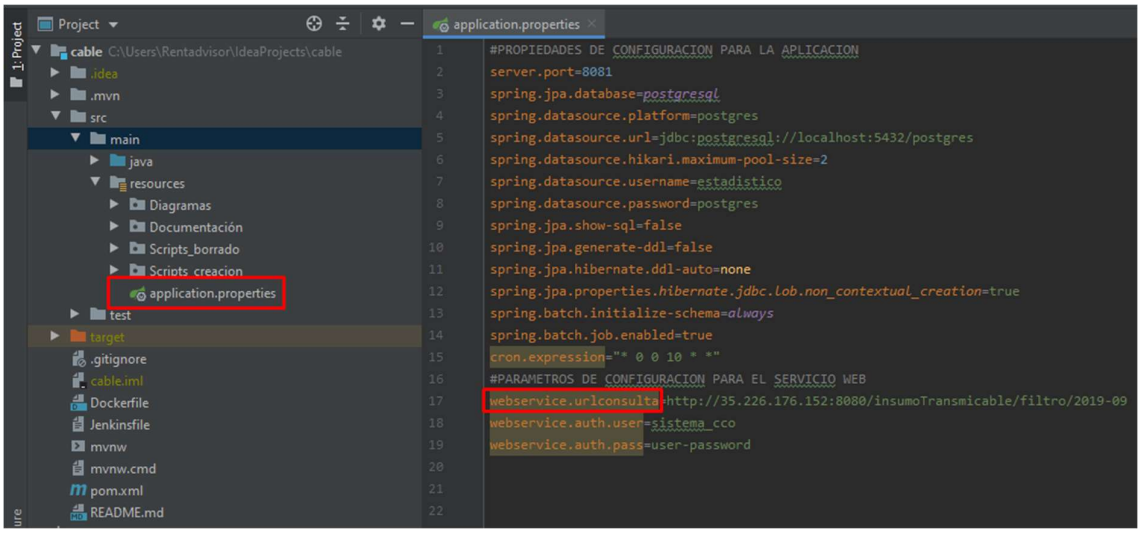
Estas deben permanecer con los valores "false" y "none" respectivamente, ya que, si se cambia su valor, se verá afectada la base de datos, por lo explicado en el inciso 3.4.

4. Servicios web

Para algunos de los módulos la información se obtiene a través del consumo de servicios web, a continuación, se entrega un listado de ellos en donde se indica el módulo al que pertenece, la propiedad a la cual debe vincularse y la url de acceso al servicio:

Módulo	Nombre propiedad*	URL del servicio
BRT	webservicebrt.fechaconsulta	http://35.226.176.152:8080/consultarIPHTroncal?fecha=20190719
Buses	webservice.fechaconsulta	http://35.226.176.152:8080/flotaOperada/2019-03-04
Cable	webservice.urlconsulta	http://35.226.176.152:8080/insumoTransmicable/filtro/2019-09
Comunicaciones	webservice.urlconsulta	https://redrecarga.rbsas.co/charge_json/?date=1552055355&uid=RBSAS&token=daf838b68e6e0449dbab22574c9487e0
Comunicaciones	webservice.auth.urisdqs	https://www.alcaldiabogota.gov.co/sdqs/servicios/RadicacionPorCanalService?wsdl

*Nombre propiedad, hace referencia como se encuentra en el application.properties del proyecto. Ejemplo para el módulo cable:



5. Especificaciones del equipo de pruebas

```
------------------
System Information
------------------
      Time of this report: 8/13/2019, 10:29:10
         Operating System: Windows 10 Pro 64-bit (10.0, Build 17763)
(17763.rs5_release.180914-1434)
                 Language: Spanish (Regional Setting: Spanish)
      System Manufacturer: LENOVO
             System Model: 80SX
                     BIOS: 0ZCN48WW (type: UEFI)
                Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz (4 CPUs),
~2.4GHz
                   Memory: 8192MB RAM
      Available OS Memory: 7902MB RAM
                Page File: 8570MB used, 7885MB available
              Windows Dir: C:\Windows
          DirectX Version: DirectX 12
      DX Setup Parameters: Not found
         User DPI Setting: 96 DPI (100 percent)
       System DPI Setting: 96 DPI (100 percent)
          DWM DPI Scaling: Disabled
                 Miracast: Available, with HDCP
Microsoft Graphics Hybrid: Not Supported
           DxDiag Version: 10.00.17763.0001 64bit Unicode
  DxDiag Previously: Crashed in Direct3D (stage 2) & DirectSound (stage 3) &
DirectInput (stage 1) & Diagnostics (stage 1). Re-running DxDiag with
"dontskip" command line parameter or choosing not to bypass information
gathering when prompted might result in DxDiag successfully obtaining this
information
```

La configuración actual del java es la configuración por defecto con la que se instaló:

```
C:\Users\Rentadvisor>java -XX:+PrintFlagsFinal -version | findstr /i "HeapSize
PermSize ThreadStackSize"
     intx CompilerThreadStackSize                  = 0
{pd product}
    uintx ErgoHeapSizeLimit                        = 0
{product}
    uintx HeapSizePerGCThread                      = 87241520
{product}
    uintx InitialHeapSize                         := 130023424
{product}
    uintx LargePageHeapSizeThreshold               = 134217728
{product}
    uintx MaxHeapSize                             := 2071986176
{product}
     intx ThreadStackSize                          = 0
{pd product}
```

```
     intx VMThreadStackSize                         = 0
{pd product}
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

IMPORTANTE: Si aparecen problemas de memoria a la hora de ejecutar el proyecto, se debe modificar la cantidad de memoria asignada a la aplicación la JVM (máquina virtual java), en el ambiente de ejecución. A continuación, se muestra un tutorial donde muestra esta asignación manualmente:

5.1 Cómo aumentar la memoria Java en Windows

En el siguiente enlace se encuentra un tutorial del procedimiento que se debe seguir para aumentar la cantidad de memoria permitida para java:

https://es.wikihow.com/aumentar-la-memoria-Java-en-Windows-7

Se recomienda -Xms4096m para asignar 4 GB de memoria a Java.

NOTA: Con una máquina de 8GB de memoria, la cantidad máxima de filas(registros) permitidas en un archivo Excel que se pueden procesar son 100.000 (Cien mil) ya que si se agregan más registros se generará un error.

6. Pruebas de ejecución con el comando mvn clean install

Para la ejecución de este comando debe contar con lo siguiente:

- Cambiar los valores que se encuentran por defecto en el application.properties en las siguientes propiedades, por los valores deseados:

Ejemplo para el caso del módulo de económica:

```
spring.jpa.database=postgresql
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.hikari.maximum-pool-size=2
spring.datasource.username=estadistico
spring.datasource.password=postgres
spring.jpa.show-sql=false
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none
```

```
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.batch.initialize-schema=always
excelpath.economica=C:\\Users\\daav20\\IdeaProjects\\economica\\src\\main\\res
ources\\Economica_DDMMAAAA.xlsx
excelpath.economica.downloaded = C:\\Users\\RentAdvisor\\Downloads\\
spring.batch.job.enabled=false
spring.jpa.open-in-view=false
webservice.urlstatus=http://35.245.140.104:8002/transmilenio-
api/estadistico/auditoria/{id}
```

- Validar que todas las tablas estén correctamente creadas en la base de datos
- Si no se ha realizado el cargue, cargar los Excel que se encuentra dentro del proyecto ya que las pruebas se realizan haciendo el conteo de los registros que se cargan desde estos archivos.

Si las tablas no se han creado o no se ha realizado el cargue de la información del módulo, ya sea que provenga de archivos, servicios web o base de datos, aparecerán errores en la ejecución del comando.

A continuación, una ejecución correcta:

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 143, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.1.1:jar (default-jar) @ economica ---
[INFO] Building jar:
C:\Users\Rentadvisor\Documents\Transmilenio\Proyectos\Economica\economica\targ
et\economica-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.1.3.RELEASE:repackage (repackage) @
economica ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ economica --
-
[INFO] Installing
C:\Users\Rentadvisor\Documents\Transmilenio\Proyectos\Economica\economica\targ
et\economica-0.0.1-SNAPSHOT.jar to
C:\Users\Rentadvisor\.m2\repository\org\transmilenio\economica\0.0.1-
SNAPSHOT\economica-0.0.1-SNAPSHOT.jar
[INFO] Installing
C:\Users\Rentadvisor\Documents\Transmilenio\Proyectos\Economica\economica\pom.
xml to C:\Users\Rentadvisor\.m2\repository\org\transmilenio\economica\0.0.1-
SNAPSHOT\economica-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
-
[INFO] BUILD SUCCESS
```

```
[INFO] ------------------------------------------------------------------------
-
[INFO] Total time: 04:54 min
[INFO] Finished at: 2019-07-18T16:34:18-05:00
[INFO] Final Memory: 56M/610M
[INFO] ------------------------------------------------------------------------
-

Process finished with exit code 0
```

NOTA: En ninguna de las tablas de los diferentes esquemas se deben realizar inserciones manuales, pues afecta el comportamiento normal del aplicativo y todos los cargues se realizan a través del spring batch y la auditoría que el mismo provee.

Para definir y validar casos deben parametrizarse desde las distintas fuentes origen acordadas).

7. Ejecución en Docker

7.1 Configuración del Dockerfile

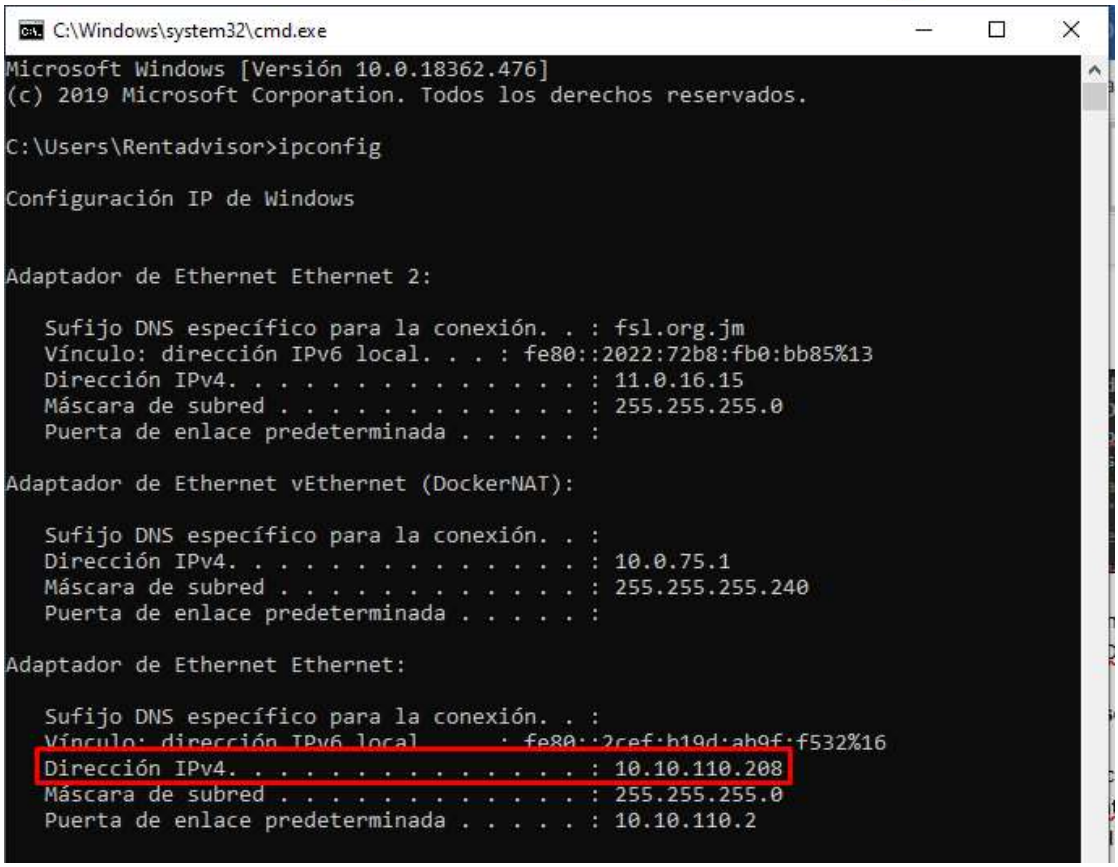
Dentro de cada proyecto se encontrará un archivo llamado “Dockerfile”, con la información básica para generar la imagen de tal manera que se logre satisfacer lo requerido para una correcta ejecución del aplicativo.

```
FROM openjdk:8-jdk-alpine
MAINTAINER ITM_COLOMBIA
ENV TZ=America/Bogota
COPY target/buses-0.0.1-SNAPSHOT.jar buses.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-
jar","/buses.jar", "--
spring.datasource.url=jdbc:postgresql://10.10.110.208:5432/postgres", "--
spring.batch.job.enabled=false"]
```

Se debe tener en cuenta que en el Dockerfile se debe de cambiar la dirección de conexión de PostgreSQL por la del Docker correspondiente:

--spring.datasource.url=jdbc:postgresql://10.10.110.208:5432/postgres

Para obtener la dirección que se debe colocar en este parámetro se debe realizar el siguiente procedimiento (Ejemplo realizado en Windows):



De manera muy parecida a la ejecución del JAR por comandos, los valores que se encuentran por defecto en el “application.properties” en las siguientes propiedades, por los valores deseados, dentro del “**ENTRYPOINT**”, esta vez colocándolos entre comillas y en caso de ser más de uno separándolos por una coma (“,”).

7.2 Ejecución del JAR en Docker

Comando (Ejemplo: módulo buses)	**Comentario**
Docker build -t buses .	El nombre que se encuentra después del -t, es el nombre que se le quiere dar a la imagen
Docker run -p 8091:8081 -t buses	Cuando se ejecuta de manera local, para realizar un mapeo entre los puertos del Docker y de la maquina anfitriona, se utiliza la siguiente propiedad: "-p 8091:8081” Donde el “8091” es el puerto de la maquina local Y el “8081” es el puerto que se está utilizando en Docker

8. Proceso de solicitud POST

Para ejecutar el módulo pasando las solicitudes a través del servicio POST, se debe enviar una petición por medio de un aplicativo destinado a estos fines (por ejemplo, POSTMAN), el cuerpo de la solicitud se debe realizar así:

URL de la petición:	**Cuerpo de la solicitud**
Ejemplo para el módulo de buses:	
http://localhost:8091/cargueBuses	{ "id":"4", "url":"https://storage.googleapis.com/tmestadistico/BUSES/Buses_Flota_ComponenteProvisional_DDMMAAAA.xlsx?GoogleAccessId=375299273628-compute@developer.gserviceaccount.com&Expires=2435087071&Signature=WI5%2BtP7dNn6QSLv%2BDDCzPk4%2FfAz%2BNcwennqMZJ7jqC9wViGGT9zTZs4mrdcEm1SS2ChmUyWJmGkHN05%2FSvuJQTbkBYj21yh2%2FVA3RWDqaumq80XWxyv%2B2lT3uPEHKmFkWyhY6tZAEMGZ0eafzgMMpUGtdMLG3pcMp6CxfRv6STLYSeN%2BvEvEcI5BTm5cefzdy79ciDtZjq8HI7drUPVPu85f9ONkdsY8tKp5Vkg274FBh4xyLgtKY62pRQhHiId4mNkZPEamsMORbyKROqicJfZmWvHItDExdNomCBODVI886XEbiDe7z9tmJujZWc4yws2zNlKJN8%2FzJiz3PXMeOA%3D%3D", "job":"jobFlotaProvisional" }



Si se está realizando la ejecución desde el Docker la URL de la petición corresponde al primer puerto asignado en el comando run "-p 8091:8081", para este caso 8091

Ejemplo de la Url de la petición, para el caso de buses cuando en el application.properties se encuentra la propiedad “server.port=8081” y cuando se corre con el comando run "-p 8091:8081”: http://localhost:8091/cargueBuses

Se modifica la propiedad "spring.batch.job.enabled", si se coloca está en **TRUE** se omitirá el uso del servicio POST para dejar en espera el aplicativo y ejecutarán inmediatamente todos los Jobs del módulo tomando los archivos que se encuentre en la ruta de resources/Excels, sin embargo, si se deja en **FALSE** hará el procedimiento normal con el servicio POST, es decir, se debe realizar una petición POST (con los parámetros a indicados previamente).

8.1 Lista de los servicios por módulo

Módulo	Dirección Servicio
BRT	http://localhost:8090/cargueBrt
Buses	http://localhost:8091/cargueBuses
Cable	Funcionalidad de ejecución por modelo "cron"
Comunicaciones	Funcionalidad de ejecución por modelo "cron"
Económica	http://localhost:8092/cargueEconomica
Recaudo	http://localhost:8093/cargueRecaudo
Seguridad accidentes	http://localhost:8094/cargueAccidentes
Seguridad y convivencia	http://localhost:8094/cargueConvivencia
Técnica	http://localhost:8095/cargueTecnica
Técnica infraestructura	http://localhost:8096/cargueInfraestructura
Técnica Modos	http://localhost:8097/cargueTecnicaModos

Observación: La URL de cada módulo puede variar dependiendo de qué puerto se asigna al ejecutar el JAR con el comando Docker como se especifica en la sección 7.2 Ejecución del JAR en Docker, adicionalmente, para los módulos que indican

Funcionalidad de ejecución por modelo "cron" se hace referencia a que el módulo se ejecutará cada cierto periodo de tiempo según lo parametrizado en el application.properties, proceso el cual será detallado más adelante.

8.2 Nomenclatura de los Jobs

Para que los archivos sean descargados y procesados de manera correcta por cada uno de sus módulos, es importante tener claro cuál job se debe ejecutar en el JSON de la solicitud POST, es decir, para prevenir que se dispare el job de kilometraje programado cargando un archivo de flota vinculada (por ejemplo), a continuación, se nombran todos los Jobs del módulo buses:

- jobKilometrajeProgramado
- jobKilometrajeEjecutado
- jobFlotaVinculada
- jobFlotaProgramada
- jobFlotaProvisional
- jobFlotaInmovilizada
- jobVelocidadBuses*
- cuboIkbZonal
- jobFlotaOperada

Nota: El job de "velocidad buses" se ejecuta una vez sea finalizado el job de "flota vinculada" teniendo en cuenta la dependencia que existe de los buses cargados para estos dos módulos.

Adicional a esto, el nombre del job debe pasarse en el tercer parámetro del JSON que se mencionó en el inciso anterior (Proceso de solicitud POST), este valor se debe indicar en el argumento "job:", para ello se puede basar en el ejemplo que ya se encuentra en el punto referenciado.

8.3 Funcionalidad de ejecución por modelo "cron"

Para los submódulos que se obtienen a través de un servicio web, se programó la ejecución de estos para que se ejecuten en una fecha u hora previamente asignada.

Para la programación se cuenta con las siguientes propiedades:

(

s: segundos

mm: minutos

hh: hora

dd: día

AA: año

)

Cuando se coloca un "*" en cualquiera de los valores significa sin importar que valor tenga este se ejecutara con el de los valores que si están especificados.

Ejemplo para programar una ejecución diaria a las 11:50 a.m., teniendo en cuenta el orden a continuación presentado:

"s mm hh dd AA"

Se vería de la siguiente forma:

"* 50 11 * * *"

9. Contenido de las carpetas:

En este numeral, se detalla el contenido de cada una de las carpetas que se encuentran dentro del directorio de resources.

9.1 Diagramas:

En esta carpeta se encuentran los diagramas de clases de stage, DWH y cubos correspondientes al módulo del proyecto.

Ejemplo, módulo de económica:

- Diagramas/economicacub.ucls
- Diagramas/economicadwh.ucls
- Diagramas/economicastg.ucls

9.2 Documentación:

En esta carpeta se encuentran la documentación correspondiente al módulo del proyecto.

Ejemplo, módulo de buses:

- Documentación/REQ001-Definicion Stage Buses v1.docx
- Documentación/REQ001-Definicion DWH Buses v1.docx
- Documentación/Mensajes de error.docx

9.3 Excels:

En esta carpeta se encuentran los archivos correspondientes al módulo del proyecto, utilizados para realizar los cargues de pruebas y con los que se debe realizar el cargue para el correcto funcionamiento de las pruebas.

Ejemplo, módulo de buses:

- Excels/Buses_Kilometraje_KmEjecutado.xlsx
- Excels/Buses_Kilometraje_KmEjecutado_FUNCIONAL.xlsx
- Excels/Buses_Kilometraje_KmProgramado.xlsx
- Excels/Buses_Kilometraje_KmProgramado_FUNCIONAL.xlsx
- Excels/flotaprogramada.xlsx
- Excels/flotaprovisional.xlsx
- Excels/flotavinculada.xlsx
- Excels/inmovilizados.xlsx

9.4 Scripts_borrado:

En esta carpeta se encuentran los scripts requeridos para borrar las tablas del módulo correspondiente al proyecto, en algunas ocasiones aparecerán “comunes”, por ejemplo: “drop_tablas_catalogos_dwh_comunes.sql” lo que significa que existen tablas que están presentes en más de un módulo y que al ejecutar estos scripts se verán afectados los módulos donde también se encuentren estas.

Ejemplo, módulo de económica:

- Scripts_borrado/drop_tablas_stage.sql
- Scripts_borrado/drop_tablas_hechos_dwh.sql
- Scripts_borrado/drop_tablas_catalogos_dwh_economica.sql

- Scripts_borrado/drop_tablas_catalogos_dwh_comunes.sql (La ejecución de este script, afectará a otros módulos, por ejemplo, buses)
- Scripts_borrado/drop_tablas_hechos_cubo.sql
- Scripts_borrado/drop_tablas_dim_cubo.sql (La ejecución de este script, afectará a otros módulos, por ejemplo, buses)
- Scripts_borrado/drop_schemas.sql

9.5 Scripts_consulta:

Estos son scripts de consulta interna y no deben de ser ejecutados o MODIFICADOS, contienen las consultas requeridas para poblar las tablas de los cubos.

9.6 Scripts_creación:

En esta carpeta se encuentran los scripts requeridos para crear las tablas requeridas por el módulo, en caso de que existan las tablas porque ya se han creado durante la ejecución de otro módulo, el script ignora el comando de ejecución y continúa creando las tablas faltantes.

Ejemplo, módulo de buses:

- Scripts_creacion/buses_stage.sql
- Scripts_creacion/buses_dwh.sql
- Scripts_creacion/buses_cubo.sql
- Scripts_creacion/schemas.sql
- Scripts_creacion/usuario_estadistico.sql

9.7 application.properties:

Es el archivo donde se encuentran las propiedades necesarias para correr la aplicación, estas se pueden modificar directamente en este archivo, durante la ejecución del JAR, ya sea por la línea de comando, por fichero con extensión “.bat” o en el Dockerfile.