

Proceso de realización de los ejercicios con MuleSoft realizado por Luis David Estrella García.

Antes de comenzar con los ejercicios, se hizo la instalación del software AnypointStudio de su página oficial como lo menciona el documento de ejercicios a realizar, así como usar Postman para las pruebas REST, por último, se creó un usuario en Anypoint Platform para poder desplegar las aplicaciones que sean requeridas en los ejercicios.

Ejercicio 1

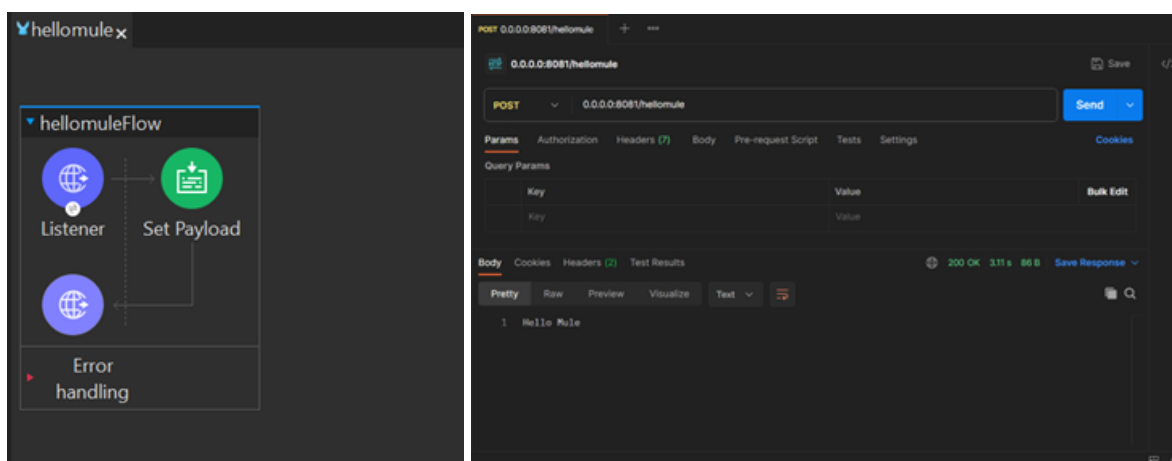
En el ejercicio 1 se comenzó por conocer la herramienta, y empecé por crear un proyecto llamado HelloMule, y una vez creado el proyecto conocí su interfaz, el cual contenía explorador de archivos en el cual están los elementos que se crean al iniciar el proyecto, además de una paleta de herramientas que se pueden usar, por último un apartado terminal y secciones específicas para poder modificar o visualizar diferentes configuraciones y logs mostrados por el software.

Después de agregar de la paleta los módulos http y set payload al área de trabajo, se configuró el puerto asignando los valores por defecto el conector de http y asignando nombre en la ruta con el mismo nombre, para posteriormente escucharlos en las pruebas con REST a través de Postman. Después de agregar el módulo http se agregó el módulo Set Payload, que básicamente es una carga de información, en el cual lo que mandamos es un texto sencillo Hello Mule.

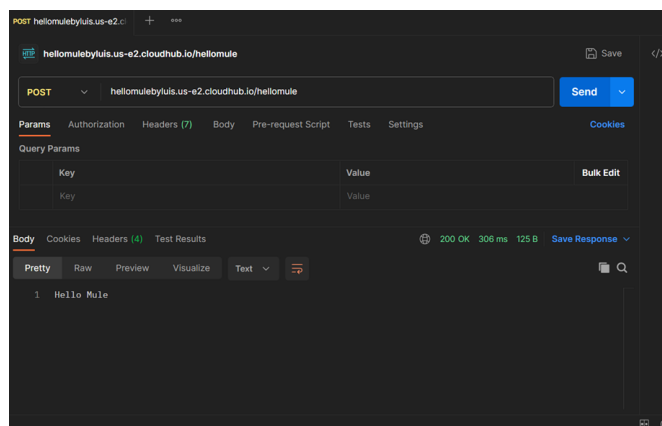
Después de tener la primera aplicación, se guardan los cambios y en el área de trabajo se da click derecho y corremos el proyecto, esperamos que compile hasta que nos aparezca Deployed en la terminal del software, cuando termina quiere decir que se ejecutó correctamente. Posteriormente se abre Postman nuestro software REST, para hacer una petición POST, donde escribimos la ruta 0.0.0.0:8081/hellomule el cual son los puertos antes configurados en el módulo http, enviamos la petición y notaremos que nos responde un 200 OK, que nos indica que se envió correctamente y la respuesta nos contesta con un Hello Mule, que es el texto ingresado en el módulo Set Payload.

Anexo evidencia de capturas de pantalla de la aplicación.

```
<terminated> hellomule [Mule Applications] Mule Server 4.5.1 EE (Terminated 12 nov 2023 15:14:04) [pid: 14488]
*****
INFO 2023-11-12 15:12:45,115 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector
INFO 2023-11-12 15:12:45,123 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
*****
* - - + DOMAIN + - - * - - + STATUS + - - *
* default * DEPLOYED *
*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
* hellomule * default * DEPLOYED *
*****
```



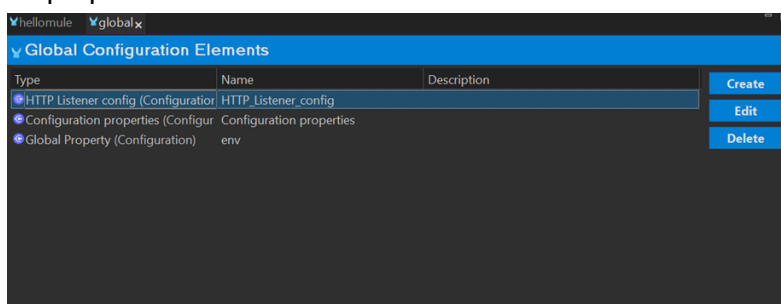
Después de compilarlo localmente, guardamos cambios hechos y exportamos el programa en extensión JAR, para posteriormente subirlo y deployar en Anypoint Platform, una vez que se sube el programa y empieza a correr correctamente nos vamos al Postman a probar que esté respondiendo, usando la dirección para hacer el POST `hellomulebyhuls.us-e2.cloudhub.io/hellomule`, y vemos que nos responde con un 200 OK, por lo que comprobamos que respondió correctamente.



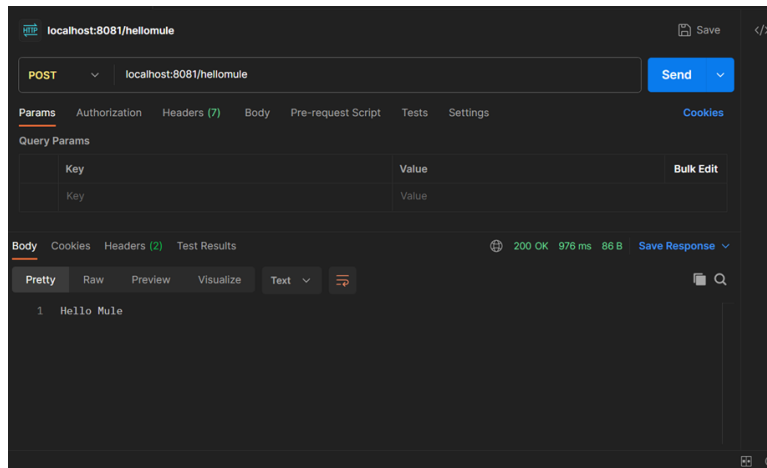
Ejercicio 2

Para empezar el ejercicio 2, empecé creando un archivo global el cual nos ayudará a gestionar mejor los proyectos en caso de que tengamos más proyectos y más robustos. El cual nos ayuda a ubicarlo rápidamente poniendo un nombre fácil de identificar, en este caso `global.xml`, en este caso queremos aplicar las variables que se encuentran en `hellomule` pasarlas a `global`, esto nos permitirá acceder a las variables cuando queramos acceder a otros proyectos. Primero cortamos la configuración `http listener` de `hellomule` que se encuentra en el editor `xml`, se guardan cambios y pegamos la configuración en los elementos del archivo `global` después le damos en guardar cambios..

Posteriormente nos ayudamos de un archivo `properties` que contendrá variables que necesitamos dependiendo del proyecto, en este caso tendremos 2, `local.properties` y `dev.properties`, en el cual el archivo `local`, se usará de manera local y el `dev` lo usaremos para usarlo en `cloudhub`, así tendremos los 2 y poder modificarlos a conveniencia. En este ejercicio guardaremos el `host` y `port`, concatenado a un `listener` que crearemos usando otra configuración de propiedades añadiéndolo en elementos globales, asignado como valor `${env}.properties`, por último para tener lista nuestra configuración debemos asignar una propiedad global el cual llamaremos `env` y nombre `local`, esto nos ayudará a hacer referencia al valor de las propiedades.



Una vez teniendo lista nuestra configuración global, podemos usarla las veces que necesitamos de esa configuración, finalmente corremos el proyecto para ver que funcione correctamente, y nos contesta con un 200 OK.



Ejercicio 3

Para el ejercicio necesitamos proteger las propiedades, entonces necesitamos crear un archivo properties que llamaremos local.secure.properties que se nos hará fácil identificar, usaremos las variables username y password, así como en local necesitamos uno para cloudhub, creamos dev.secure.properties. Siguiendo en las variables globales necesitamos agregar el módulo propiedades de configuración segura de mule que ubicamos en exchange. Después de haber agregado el modulo lo usamos el cual nos pide una configuración, asignamos nombre de archivo el cual usaremos `${env}.secure.properties`, el cual será un archivo dinámico, después en key usaremos `${secure.key}` aunque no es recomendable por buenas prácticas de seguridad donde se puede acceder o alterar facilmente, por ultimo se usa el blowfish como algoritmo de encriptación.

Posteriormente para poder registrar y leer las credenciales de una manera segura primero en nuestra área de trabajo agregamos el módulo Logger el cual necesitamos usar la función del módulo que en este caso es agregar código DataWeave en el cual agregaremos las variables Username y Password y hacemos la referencia a las variables, esto nos ayudará a que cuando se quiera acceder a estas variables primero se necesitarán descifrar haciéndolo menos vulnerable.

Con la ayuda de la herramienta Secure Properties Tool, usando una terminal donde se encuentre la ruta de la herramienta ingresamos la linea de comando:

**java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool string ** -> Uso de la herramienta

**encrypt ** -> encriptamos el valor de los archivos de nuestras propiedades

**Blowfish ** -> Algoritmo de encriptación

**CBC ** -> Ayuda a crear el código de encriptación

**MyMuleSoftKey **

"myUsernameLocal" -> Propiedad que se usa en este caso como ejemplo, pero podemos usar otro, que en nuestro caso necesitamos encriptar 4 variables lo usamos en cada uno y se crean el código como se muestra a continuación:

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Versión 10.0.22621.2506]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\barre\AnypointStudio\studio-workspace\hellomule>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePr
opertiesTool string encrypt Blowfish CBC MyMuleSoftKey "myUsernameLocal"
HbsuWJRjiubchmzQREGdsA==

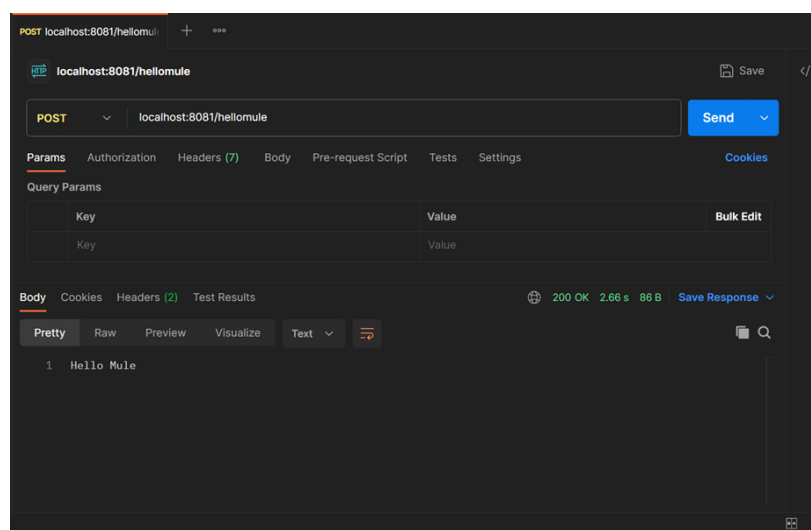
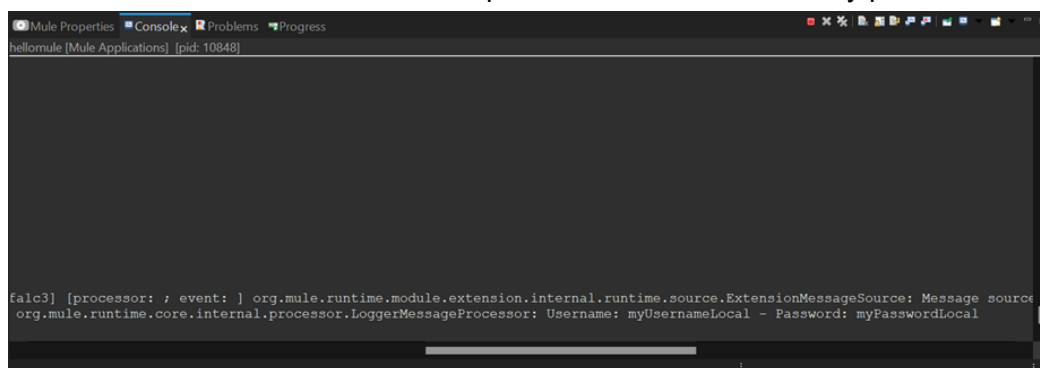
C:\Users\barre\AnypointStudio\studio-workspace\hellomule>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePr
opertiesTool string encrypt Blowfish CBC MyMuleSoftKey "myPasswordLocal"
zhoritn2db6Ud+9QuhyViQ==

C:\Users\barre\AnypointStudio\studio-workspace\hellomule>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePr
opertiesTool string encrypt Blowfish CBC MyMuleSoftKey "myUsernameDev"
HbsuWJRjiuZZis+2oogkdQ==

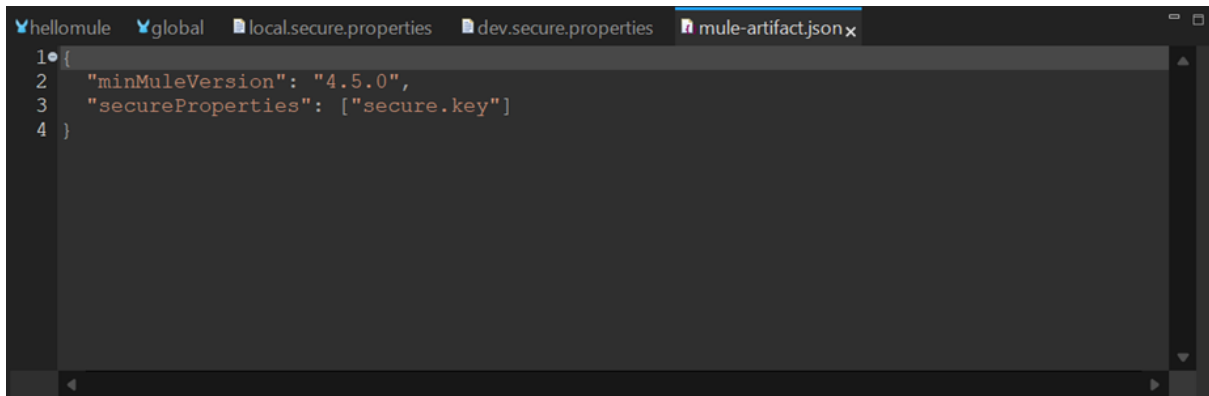
C:\Users\barre\AnypointStudio\studio-workspace\hellomule>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePr
opertiesTool string encrypt Blowfish CBC MyMuleSoftKey "myPasswordDev"
zhoritn2db6oII22158LPQ==

C:\Users\barre\AnypointStudio\studio-workspace\hellomule>
```

Posteriormente para poder probarlo en la computadora lo que se hace es ejecutar configuraciones del proyecto y agregamos una nueva variable en este caso secure.key, que es la variable que hemos creado anteriormente con el valor MyMuleSoftKey guardamos y corremos el proyecto, al hacer esto lo que no se guardan las variables, y se mandan directamente para evitar alteraciones de cualquier tipo y vemos que al correr contesta con la informacion que teniamos del username y password.

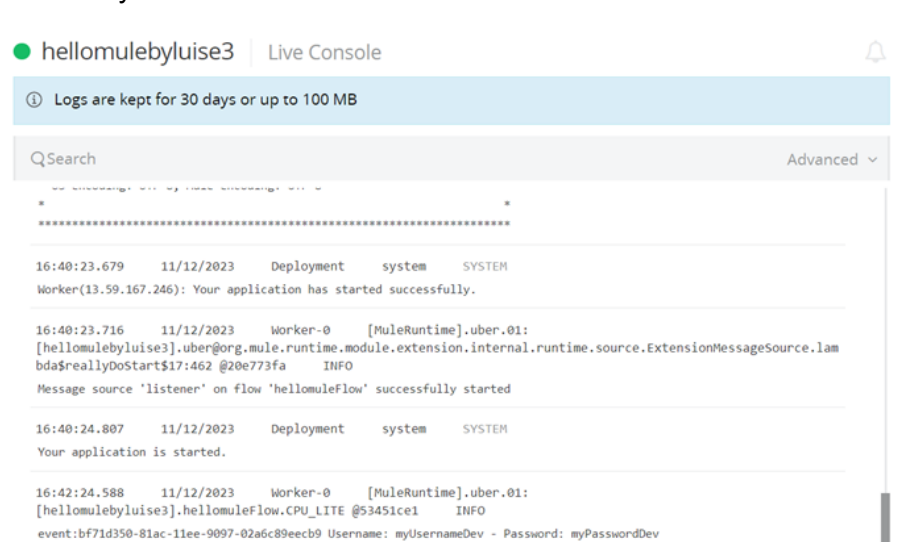


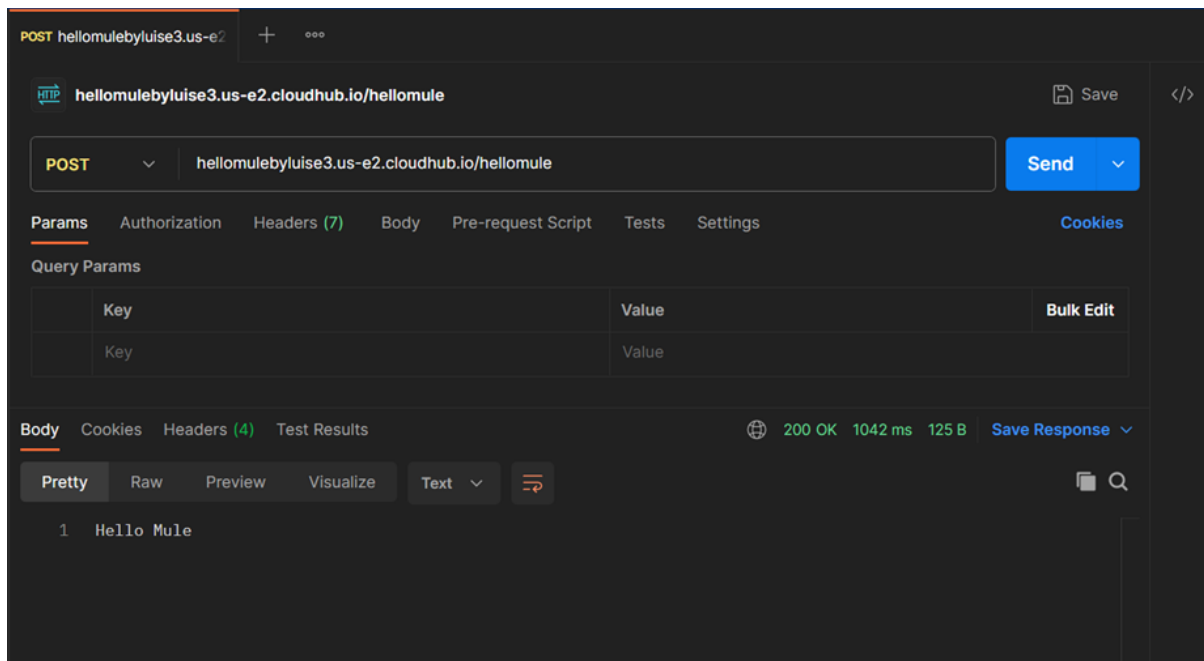
Ahora lo que buscamos es implementarlo en cloudhub con esta nueva configuración de seguridad, primero ingresamos a Anypoint platform desde la aplicación, y en la sección Runtime Manager y agregamos la configuración env y secure key que nos ayudará a referenciar en la aplicación y debemos de agregar en el archivo mule-artifact.json secureProperties que se encuentra en los archivos del proyecto, esto para ocultar el valor de secure.key.



```
1 {
2   "minMuleVersion": "4.5.0",
3   "secureProperties": ["secure.key"]
4 }
```

Finalmente deployamos a cloudHub de nuestro proyecto y esperamos a que se termine de ejecutar y comprobamos que todo salió correctamente cuando enviamos la petición en Postman y nos responde con los valores Username y Password .



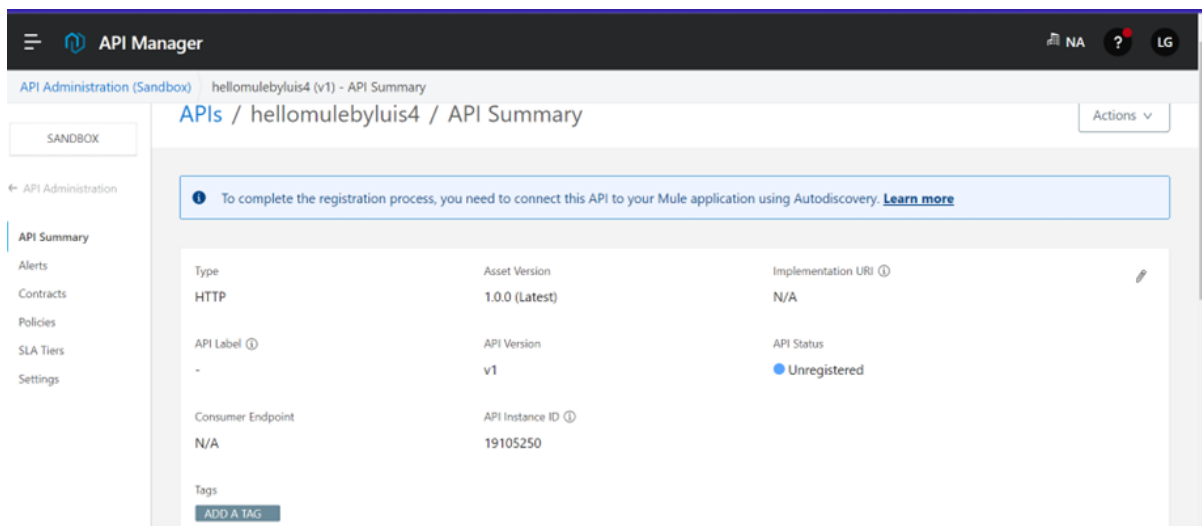


Ejercicio 4

Ahora crearemos una nueva API a través de Anypoint Platform, en el cual ingresamos y nos vamos a API Manager y una vez dentro debemos darle en Add new API, seleccionamos Mule Gateway y al seleccionar esta opción nos permitirá conectarlo con la aplicación de Anypoint Studio.

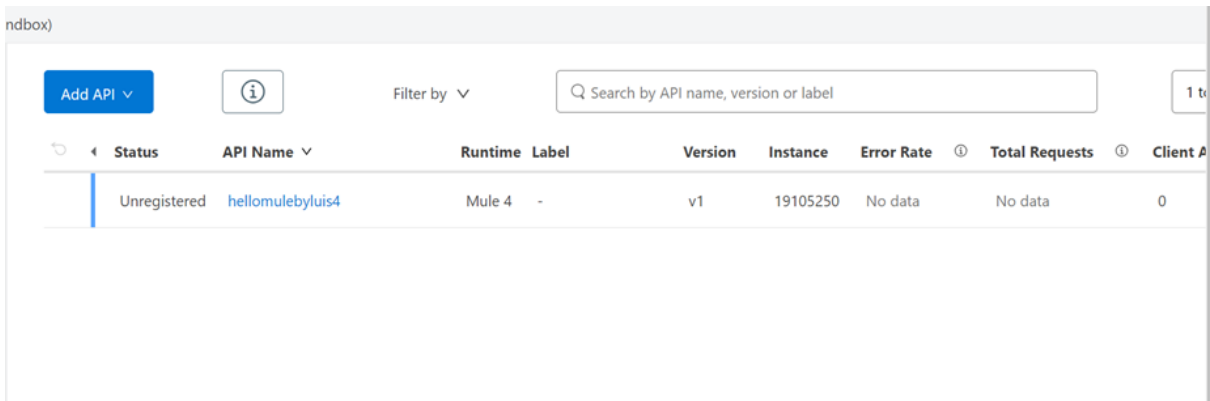
Después debemos agregar una API en Exchange, el cual debemos asignar un nombre que nos sea fácil de identificar, y seleccionamos que queremos una API HTTP ya que lo podremos usar con una URL, asignamos versión y le damos en crear.

Ya que tenemos todo listo, podemos visualizarlo de la siguiente manera, donde tenemos la información respecto a la API,



Ahora buscamos crear una propiedad en el software Anypoint Studio, vamos a nuestro programa de hellomule para agregar el api.id tanto local y dev propiedades, y le asignamos el API Instance ID, que se encuentra en el API Manager. Posteriormente para poder referenciar necesitamos irnos a global.xml y elementos globales, añadimos API Autodiscovered y ponemos el nombre \${api.id} que lo referenciamis facilmente y por último seleccionamos hellomuleFlow.

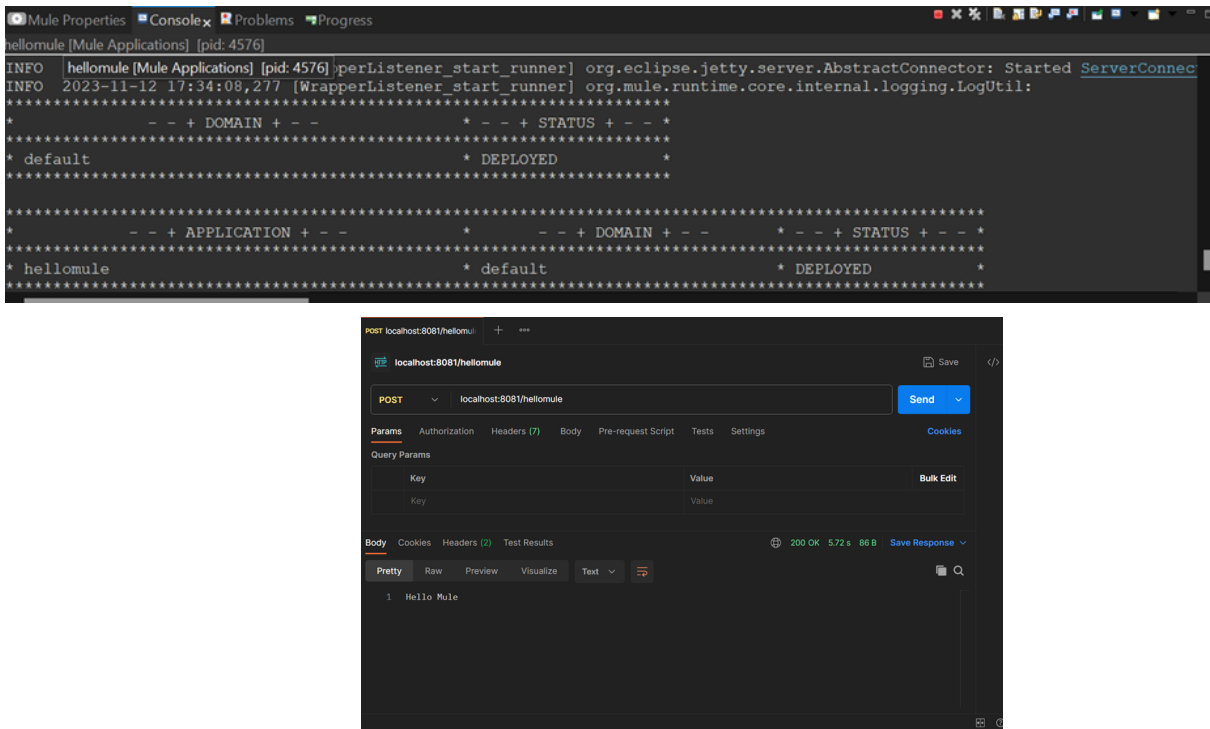
Para poder probar el programa de manera local y ver que nuestra API creada está funcionando y enlazando correctamente necesitamos el cliente id y cliente secreto que se encuentra en API Manager en la pantalla principal a lado de Add API, como se muestra a continuación.



The screenshot shows the Anypoint API Manager interface. At the top, there is a search bar and a table with columns: Status, API Name, Runtime, Label, Version, Instance, Error Rate, Total Requests, and Client ID. A single row is visible with the following data:

Status	API Name	Runtime	Label	Version	Instance	Error Rate	Total Requests	Client ID
Unregistered	hellomulebyluis4	Mule 4	-	v1	19105250	No data	No data	0

Una vez identificado las credenciales necesitamos ingresarlas en ANypoint Studio, ingresando en la pestaña de preferencias, después en Anypoint Studio y Administrador de API, dentro de la pestaña ingresamos las contraseñas correspondientes y le damos validate para verificar que son correcta, le damos en aplicar y cerrar. Para ver que siga ejecutando sin problemas, volvemos a correr el proyecto y verificamos como se muestra a continuación.

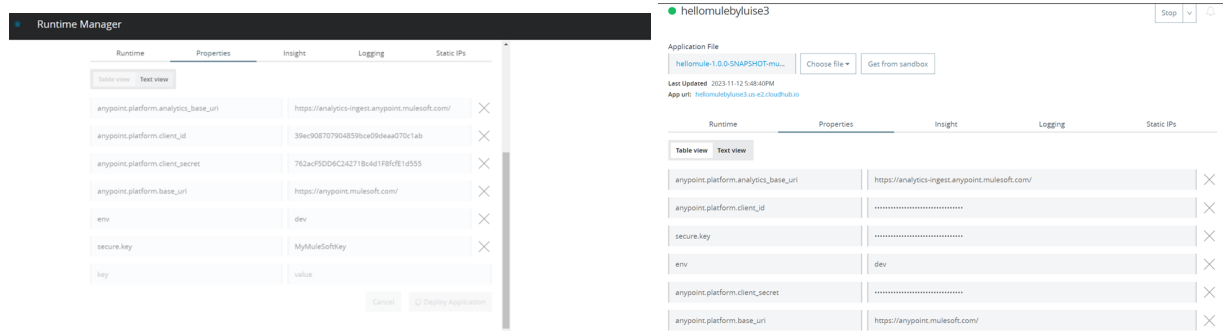


The first screenshot shows the Eclipse IDE console with the following output:

```
hellomule [Mule Applications] [pid: 4576]
INFO [hellomule [Mule Applications] [pid: 4576] perListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector
INFO 2023-11-12 17:34:08,277 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
*****
* - - + DOMAIN + - - * - - + STATUS + - - *
*****
* default * DEPLOYED *
*****
*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*****
* hellomule * default * DEPLOYED *
*****
```

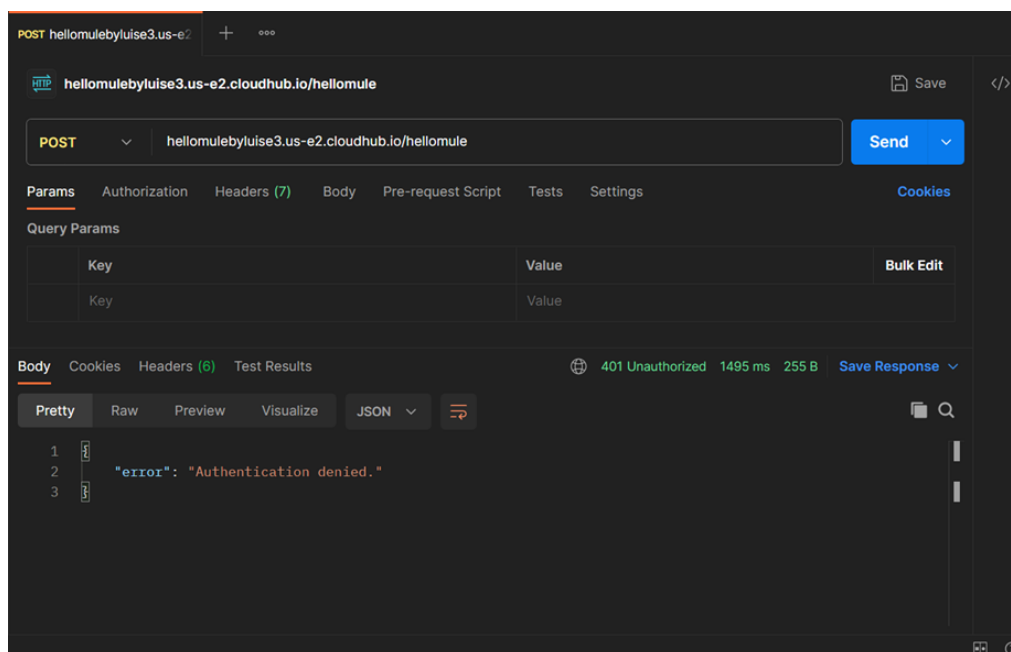
The second screenshot shows the Anypoint Studio interface with a REST client. The request is a POST to localhost:8081/hellomule. The response is a 200 OK with a body of "Hello Mule".

Ahora lo implementaremos en cloudfoundry, abrimos mule-artifact.json para poder ingresar propiedades que nos ayudan a la seguridad de la API, una vez realizado podemos deployar en cloudfoundry, seleccionamos la opción de sobrescribir el proyecto ya que no queremos muchos programas haciendo lo mismo cuando estemos realizando las pruebas, verificamos en la parte de properties que se encuentren las credenciales de la API que obtuvimos de Anypoint Studio, finalmente lo deployamos y verificamos todos los datos y el software se encuentra en ejecución como se muestra a continuación.

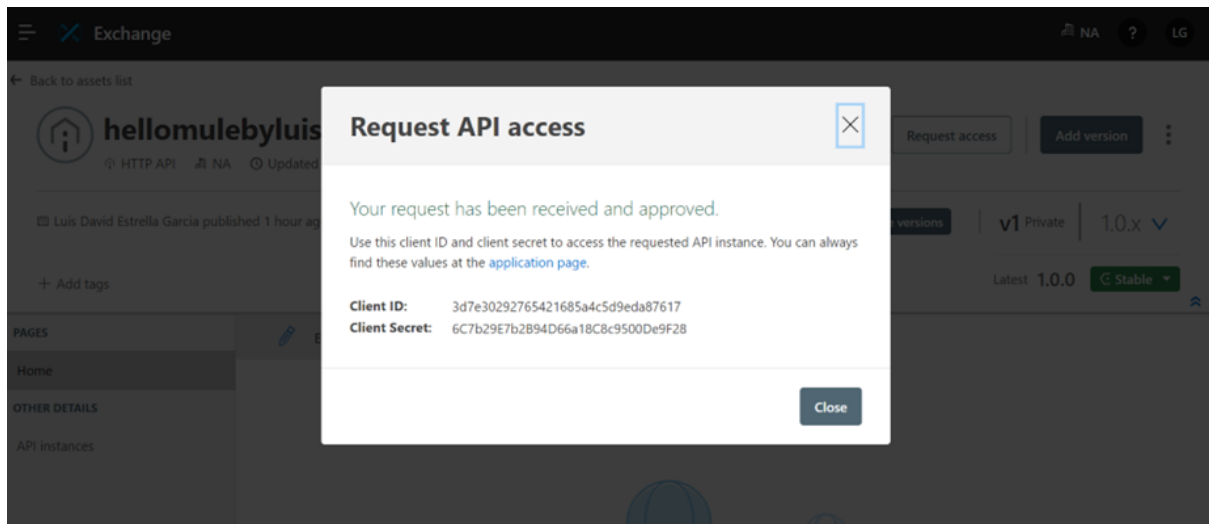


Ejercicio 5

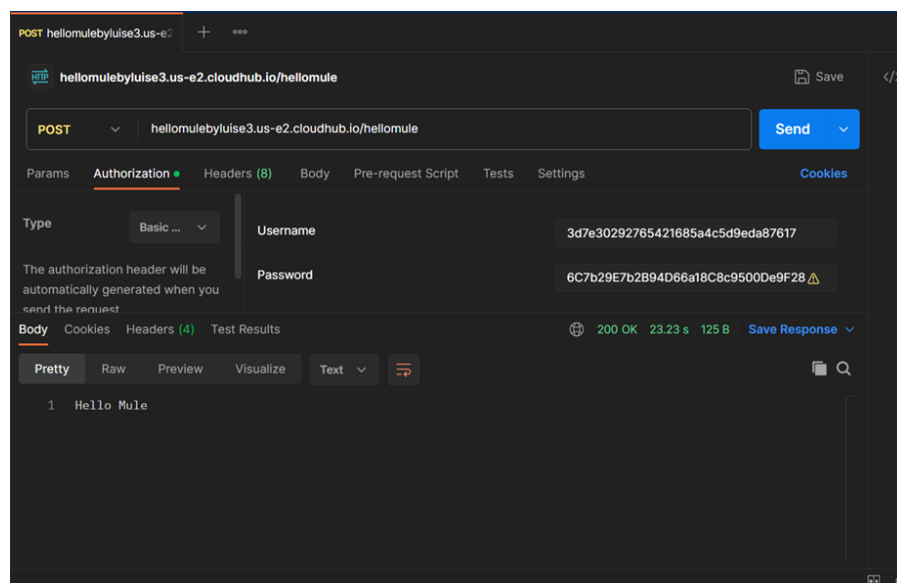
Ahora es necesario establecer políticas de cumplimiento del cliente en la API que estamos desarrollando, para ello vamos ingresamos a anypoint PPlatform, después a API Manager, y una vez dentro le damos en políticas en las opciones disponibles, buscamos la política de cumplimiento de ID de cliente seleccionamos la más reciente que es la versión que está más actualizada el cual nos ayuda a estar al día. Dejamos en default las opciones que se requieran y le damos en configurar política, Probamos la aplicación para visualizar si se aplicaron los nuevos cambios, y como vemos a continuación se hizo correctamente ya que no nos deja visualizar la información.



Para poder visualizar la información necesitamos crear las credenciales de la API, tenemos que irnos a la parte de Exchange que se encuentra en Anypoint platform, nos aparecerá nuestra API creada anteriormente, lo seleccionamos y le damos en Request API access y creamos con la opción default de la instancia de la API, al hacer esto debemos crear una nueva aplicación para que podamos generar las nuevas credenciales. Asignamos un nombre fácil de reconocer en nuestro caso hellomuleapp y le damos en create, al crear nos regresará a la parte de Request, y le damos en finalizar, esperamos a que se apruebe automáticamente, una vez creada correctamente se habrán generado las nuevas credenciales como se muestra a continuación.



Finalmente para verificar que las nuevas credenciales para acceder, es necesario que ingresemos una autorización en REST, en este caso le damos uno básico, asignamos el username con Client ID, así como password el Client Secret generados hace un momento. Mandamos la petición nuevamente y vemos que ahora nos responde con un 200 OK, ya con autenticación.



Ejercicio 6

En este apartado vemos varios apartados teóricos, el cual me ayudó a entender de mejor manera todo lo relacionado a las API, la cual nos ayudará a realizar mejores conexiones y comunicaciones entre aplicaciones, haciéndolo más simple y fácil de usar para los usuarios, ya que solo accede a las secciones o información requerida y no toda la información que muchas veces no es rentable o es difícil de procesar.

Poco a poco la importancia de tener API implementado en los sistemas ya que actúan como una guía que establece las reglas y expectativas para la construcción y comunicación de la API, ya que como desarrolladores es necesario tener formatos específicos para buenas prácticas y mejor documentación. Como lo dice en la documentación: La creación de una especificación de API le permite codificar en torno a la definición de API, en lugar de definir su API con código. Nos ayuda a no realizar código repetitivo y que vuelven más lentos los sistemas.

¿De qué manera se vuelven más prácticos?

Cuando ya no tenemos que estar especificando qué métodos estamos utilizando, simplemente realizamos la carga o petición de la variable, porque ya realizamos el diseño bien definido que nos hará los métodos cuando lo queramos en cada proyecto. Es importante saber los métodos que existen para saber que utilizar en cada ocasión, y se vuelve más sencillo desarrollar, así como los códigos de estado, que incluso podemos definir, pero es mejor usar los que ya están establecidos para mejor lectura entre otros desarrolladores y tener buena documentación.

Ejercicio 7

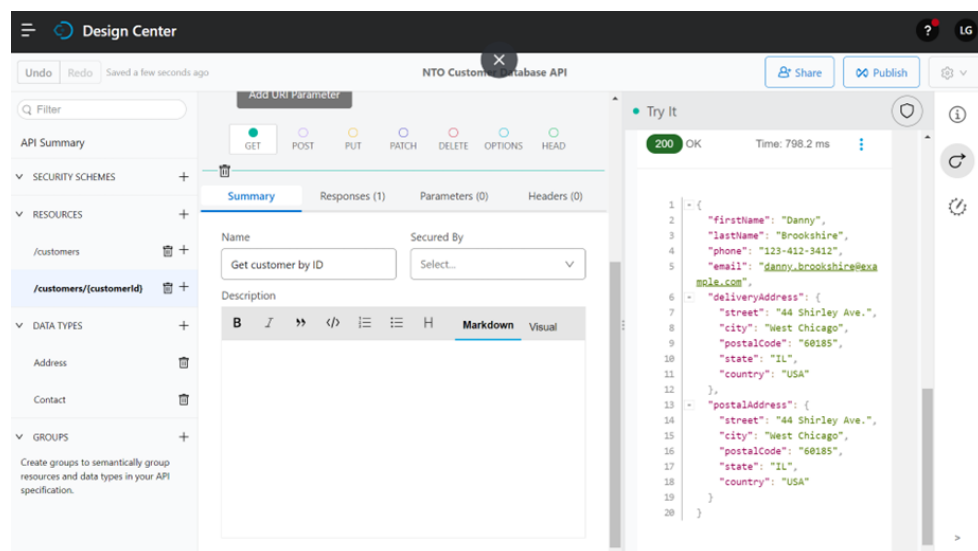
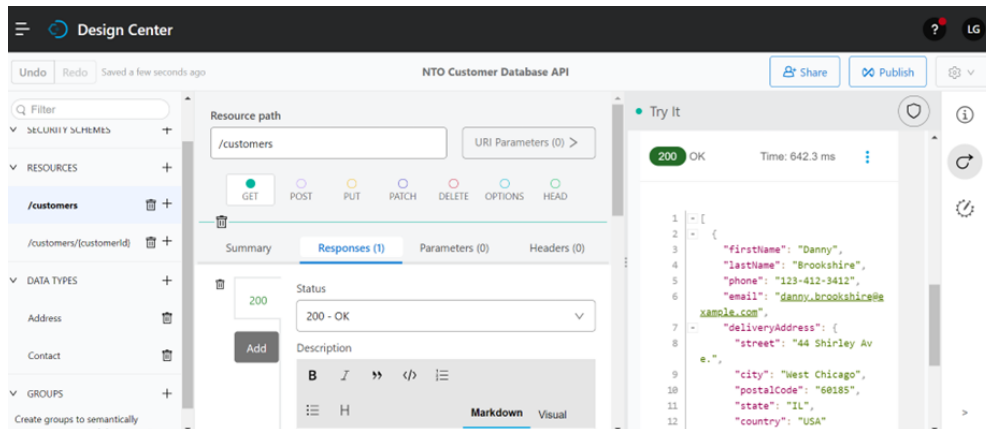
En este ejercicio creamos una especificación de API, que en pocas palabras vamos a solucionar un problema que se tiene una base de datos antigua y la actualizaremos, primero vamos a Design Center que se encuentra en Anypoint Platform, una vez ahí creamos la nueva especificación de API.

Asignamos el nombre NTO Customer Database API que es el nombre de la empresa, después asignamos los parámetros como el nombre, versiones, descripción para identificar lo que estamos actualizando.

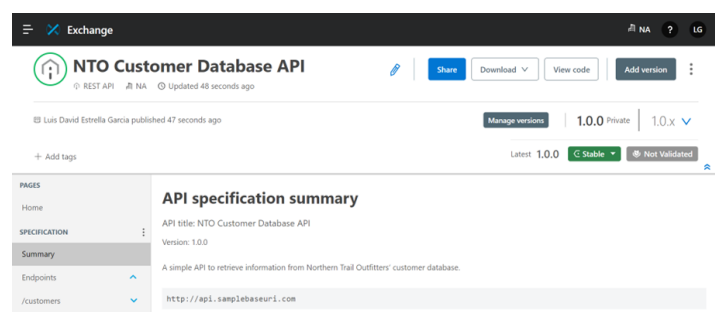
Agregamos 2 nuevos tipos de datos que en este caso son: Dirección y Contactos, en el cual debemos ponerle nombre, el tipo de dato que son, normalmente en esta parte tendremos objetos ya que guardan varias cadenas de información. Agregamos las propiedades que se definieron al momento de analizar la base de datos. Por último debemos agregar ejemplos para hacer las pruebas posteriormente y verificar que se están mandando correctamente.

Debemos agregar los recursos de especificación que nos ayudarán a definir la ruta, definir el método que usaremos y una descripción. En este caso de igual manera tenemos 2, la parte de clientes, que tendremos información de los datos que antes definimos. Agregamos las respuestas que necesitamos para definir en cada situación, en este caso solo usaremos uno de prueba que es el 200 OK, que si todo sale bien se hace la petición Get correctamente debe responder como asignemos, finalmente si queremos filtrar el cliente por ID, creamos otra especificación que quedaria asi: /customers/{customerId}, en el cual obtendremos el customerId y filtrará dependiendo el cliente.

Para probar tenemos la herramienta de Design center y nos permitirá simular la respuesta con el ejemplo que antes ingresamos, además de que se pueden hacer pruebas ya con URL. Probamos dándole en el segundo botón del menú vertical de la derecha. Tanto de customer y customer filtrado, como se muestra a continuación.



Después debemos configurar el servicio de simulación para poder publicarlo en Anypoint Exchange esto nos ayudará a que se pueda utilizar esta API en caso de que se requiera sin tener que estar haciéndolo gastando más tiempos y recursos, En la parte de arriba derecha, encontramos el botón de Publish, lo seleccionamos para publicarlo en Exchange, asignamos la versión 1.0.0 en caso de ser la primera o asignar la que corresponda y le damos en publicar y así los usuario que requieran usarlo, podrán visualizarlo.



Ejercicio 8

Se comienza verificando si se tiene instalado una versión de jdk y para ser mas especifica la versión de jdk 8 ya que al principio parecía que todo estaba bien pero al ir avanzando y al estar leyendo me di cuenta que se necesitaba la versión de jdk 8 para que fuera más estable y saliera sin errores, si no se tenia configurado y tampoco instalado como fue mi caso se tiene que descargar el jdk e instalar (es un ejecutable .exe en el caso de windows) y después se tiene que ir al disco duro principal que también es llamado C: después a todos los archivos, java y buscamos el jdk que acabamos de instalar y nos vamos a la carpeta bin que está dentro del jdk, seleccionamos la ruta la cual quedaría algo así C:\todoslosarchivos\Java\jdk8.0.0\bin.

Después en el buscador de windows escribimos la palabra env y le damos a la primera opción que nos aparece y nos debería de mostrar las variables de entorno en la parte de arriba le damos a la opción de path y le damos en edit nos abrirá una ventana nueva y le damos en new o nueva y pegamos la ruta y ya tenemos habilitada la opción de keytool en la terminal.

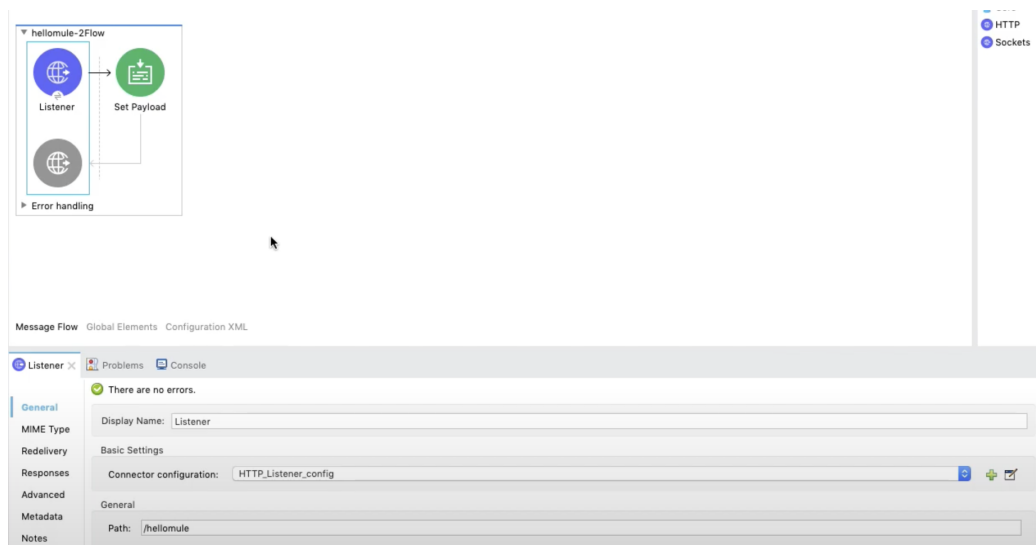
Abrimos la terminal y pegamos el siguiente comando

```
C:\Users\barre\OneDrive\Escritorio>keytool -genkeypair -alias mule -keystore keystore-hellomule.jks -keypass password -storepass password -keyalg RSA -sigalg SHA1withRSA -keysize 2048
```

Donde se pone -genkeypair el cual no se que significa pero al no ponerlo manda error, el -alias donde su como su nombre lo indica pondremos un alias de la claves, -keystore es el nombre que se le da a las llaves, el keypass y storepass son las contraseñas que debe de llevar para cifrar y descifrar las llaves y los demas codigos ya son para la forma de desencriptar el tamaño de la llave, entre otros.

Ahora nos vamos a la Herramienta y buscaremos un http listener y lo pondremos del lado izquierdo y del lado derecho pondremos un set Payload.

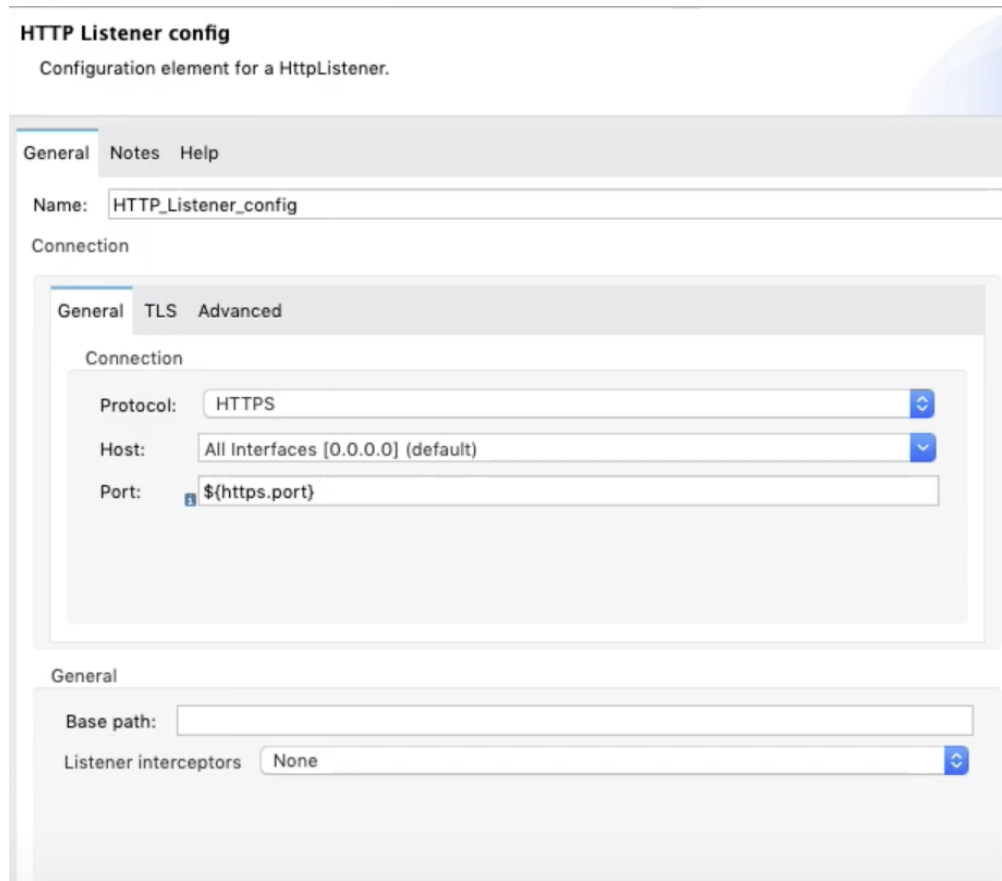
Al listener le pondremos un Conector que configuraremos mas adelante y en la parte del path le pondremos una diagonal y un nombre que queramos en mi caso es /hellomule y en el set Payload en la parte de value quitamos la opcion de formula y ponemos un mensaje para saber que ya se si se conecto de forma correcta como se muestra a continuación.



en la ruta de src/main/resource creamos un archivo llamado local.properties y adentro le pondremos https.port=8082

Ahora si editamos el http listener config que habíamos creado antes y lo configuramos de la siguiente manera:

en el protocolo le dejamos HTTPS en el host lo dejamos igual como viene y en el Port lo dejamos \${https.port}



HTTP Listener config
Configuration element for a HttpListener.

General Notes Help

Name: HTTP_Listener_config

Connection

General TLS Advanced

Connection

Protocol: HTTPS

Host: All Interfaces [0.0.0.0] (default)

Port: \${https.port}

General

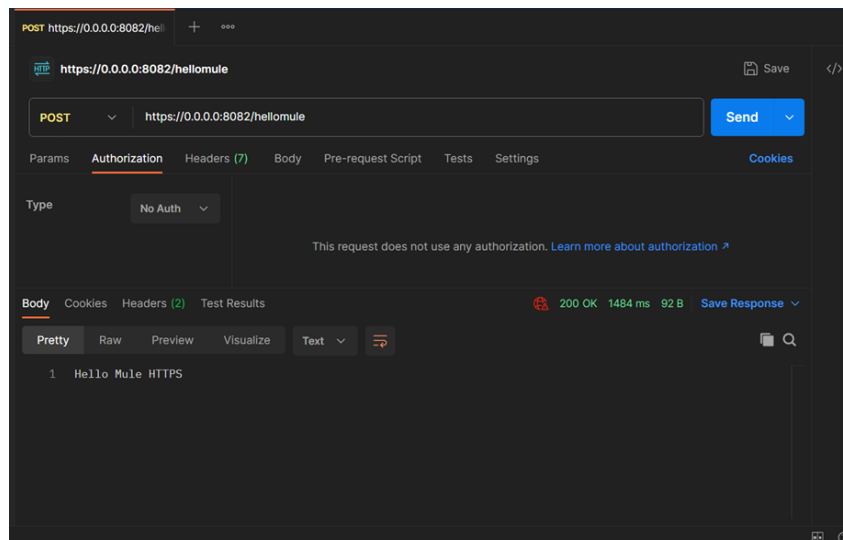
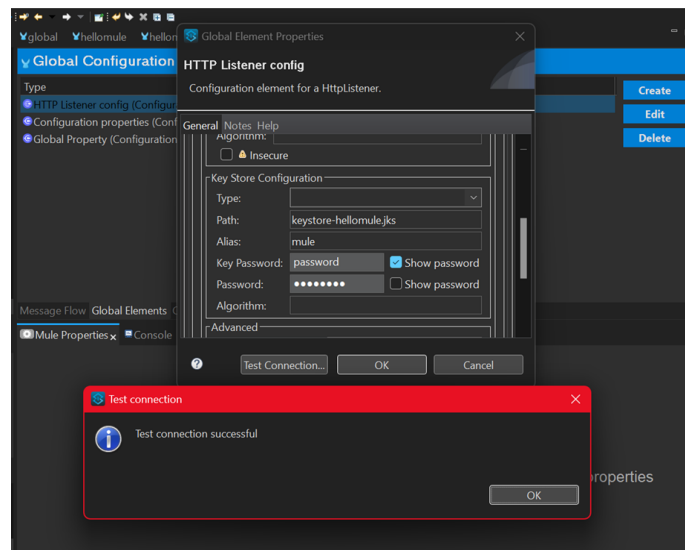
Base path:

Listener interceptors: None

Nos vamos a global Element y creamos un configuration properties donde en file ponemos local.properties así como llamaos al archivo que creamos anteriormente.

ahora arrastramos el archivo que se generó con el código en cmd a la ruta de src/main/resource para poder utilizarlo.

Abrimos el HTTP Listener Config y nos vamos a la parte de TLS y le damos a init inline para configurarlo, nos dirigimos hasta la parte de Key Store Configuration vamos llenando los campos dependiendo de lo que hayamos puesto en el comando de cmd en la parte de Path ponemos el nombre que pusimos en -keystore, en alias lo que pusimos en -alias en key password y en Password ponemos lo que pusimos en keypass y en password respectivamente y para confirmar que está listo y funcionando le damos en test connection y debe de salir bien. Por último en postman ponemos Post <https://0.0.0.0:8082/hellmule>



Ejercicio 9

Para este ejercicio es necesario instalar Maven en caso de no tenerlo instalado y una vez instalado lo que hice fue configurar el archivo settings.xml en mi disco local c con la configuración recomendada, así como ir verificando las versiones requeridas. Después configura Maven en Anypoint Studio, en el cual lo hice ingresando a las preferencias del software y anypoint Studio, seleccionando Override user settings path, y seleccionando el archivo que cree, añadiendo la configuración del archivo.xml.

Después cree un nuevo proyecto asignando una carga básica al Listener. Después configuré el pom.xml que se encuentra en la carpeta del proyecto, esto para ejecutar maven con las versiones que requiere la aplicación. Posteriormente en la carpeta del proyecto ejecuté mvn clean package deploy -DmuleDeploy, el cual ejecutará el proyecto con Maven, sin embargo, me dió un error que no pude solucionar, reintentando con otras configuraciones que sin tener éxito. Después se harán otras pruebas. Anexo las capturas.

```

C:\Windows\System32\cmd.exe
[INFO] --- mule:4.0.0:generate-test-resources (default-generate-test-resources) @ hellomulev2 ---
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ hellomulev2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- mule:4.0.0:test-compile (default-test-compile) @ hellomulev2 ---
[INFO] --- surefire:3.1.2:test (default-test) @ hellomulev2 ---
[INFO] --- mule:4.0.0:package (default-package) @ hellomulev2 ---
[INFO] Building zip: C:\Users\barre\AnypointStudio\studio-workspace\hellomulev2\target\hellomulev2-1.0.0-SNAPSHOT-mule-application.jar
[INFO] --- mule:4.0.0:verify (default-verify) @ hellomulev2 ---
[INFO] --- install:3.1.1:install (default-install) @ hellomulev2 ---
[INFO] Installing C:\Users\barre\AnypointStudio\studio-workspace\hellomulev2\pom.xml to C:\Users\barre\.m2\repository\com\mycompany\hellomulev2\1.0.0-SNAPSHOT\hellomulev2-1.0.0-SNAPSHOT.pom
[INFO] Installing C:\Users\barre\AnypointStudio\studio-workspace\hellomulev2\target\hellomulev2-1.0.0-SNAPSHOT-mule-application.jar to C:\Users\barre\.m2\repository\com\mycompany\hellomulev2\1.0.0-SNAPSHOT\hellomulev2-1.0.0-SNAPSHOT-mule-application.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:41 min
[INFO] Finished at: 2023-11-12T23:23:38-06:00
[INFO] -----
C:\Users\barre\AnypointStudio\studio-workspace\hellomulev2>

```

```

C:\Windows\System32\cmd.exe
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ hellomulev2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- mule:4.0.0:test-compile (default-test-compile) @ hellomulev2 ---
[INFO] --- surefire:3.1.2:test (default-test) @ hellomulev2 ---
[INFO] Skipping execution of surefire because it has already been run for this configuration
[INFO] --- mule:4.0.0:package (default-package) @ hellomulev2 ---
[INFO] --- mule:4.0.0:verify (default-verify) @ hellomulev2 ---
[INFO] --- install:3.1.1:install (default-install) @ hellomulev2 ---
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 58.278 s
[INFO] Finished at: 2023-11-13T00:34:37-06:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-install-plugin:3.1.1:install (default-install) on project hellomulev2: The packaging for this project did not assign a file to the build artifact -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
C:\Users\barre\AnypointStudio\studio-workspace\hellomulev2>

```

Conclusión final

Aunque enfrenté algunas dificultades durante los ejercicios con la herramienta MuleSoft, disfruté explorar sus funcionalidades y abordar los desafíos que surgieron. La experiencia me permitió aprender y mejorar mis habilidades, y conforme avanzaba en el desarrollo, pude superar las dificultades mediante resolución de problemas. En esta práctica enfrenté varios obstáculos y los fui solucionando a medida que avanzaba en el proyecto, siendo la última con mayores dificultades.