# Descripción de la aplicación.

La aplicación fue desarrollada en Java 7, con una base de datos MySql y se utilizó el servidor Payara Micro

El proyecto fue creado en Maven, y para los servicios rest se utiliza la librería Jersey

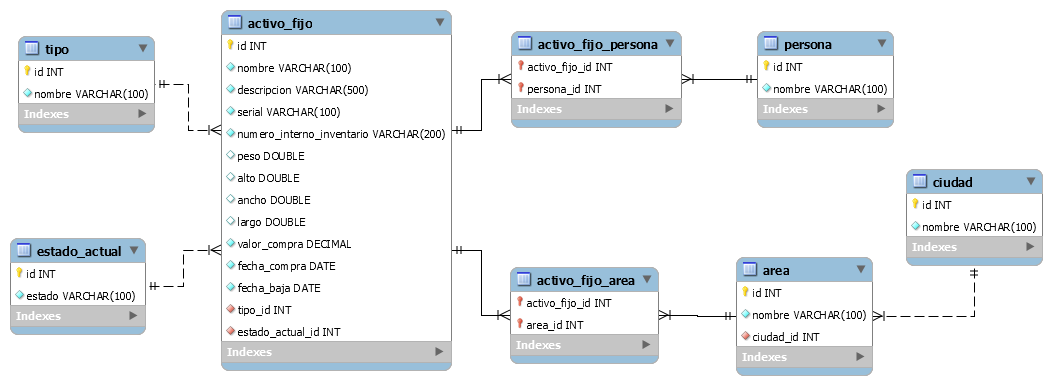
El proyecto se divide en tres capas una capa web (Clase ServicioActivos en el paquete com.asd.api.rest.activos.servicios) en donde se implementan los servicios REST, una capa de control (Clase ActivosBean en el paquete com.asd.api.ejb.activos.impl) implementada en un ejb en la cual se controlan las transacciones y alguna lógica de negocio y una capa de persistencia implementada con jpa.

Se entrega archivo “base\_de\_datos.sql” para la instalación de la base de datos y archivo de despliegue “asdapi-web-1.0.war”.

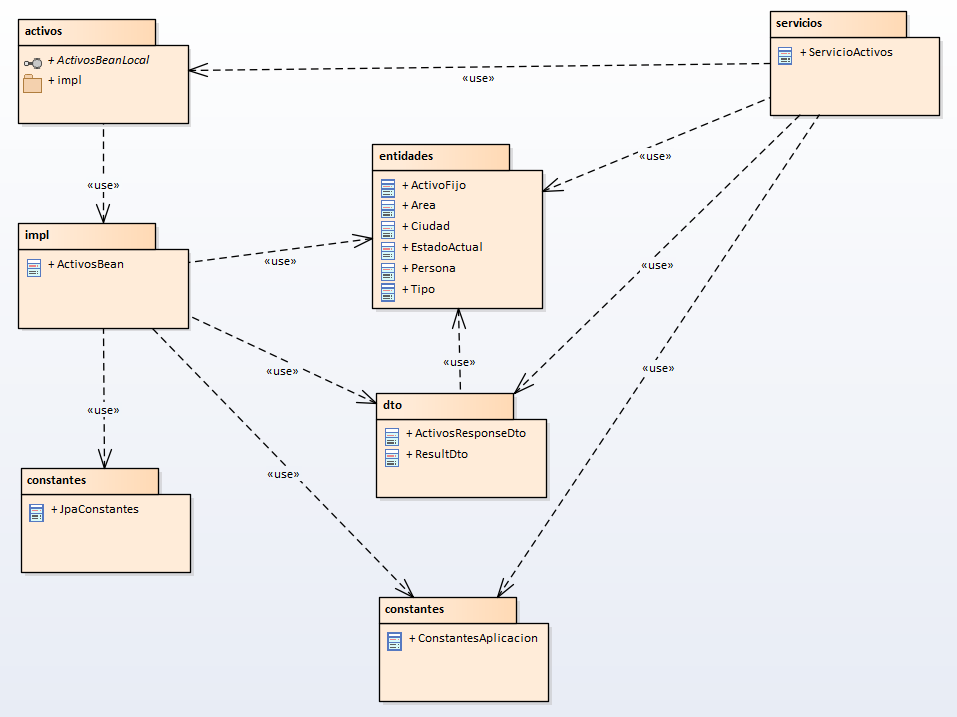
Para la generación del war se debe abrir el proyecto en netbeans o eclipse con sus respectivas funcionalidades maven activadas y se debe compilar el proyecto maven asdapi, luego se bebe localizar el archivo desplegable en la carpeta “asdapi-web\target”

Las pruebas del servicio se entregan en el archivo “ASD\_API.postman\_collection.json”

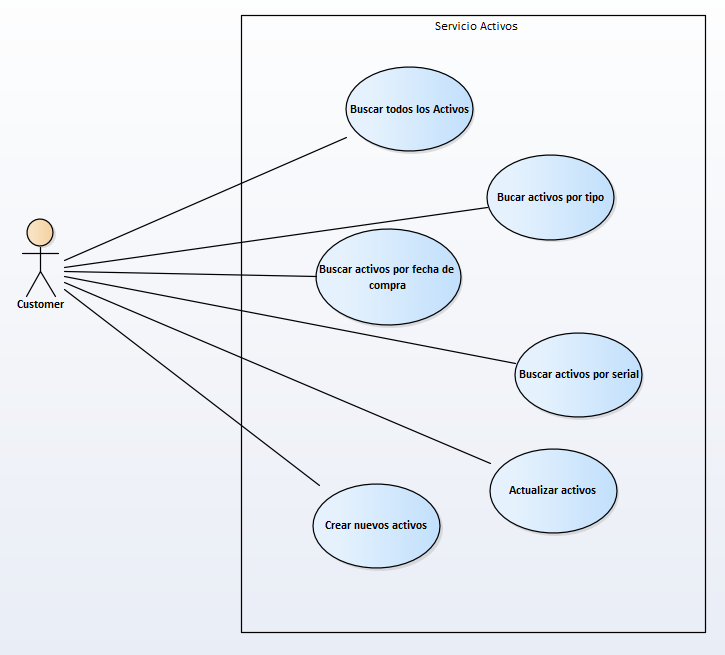
# Modelo de Datos



# Modelo de Paquetes



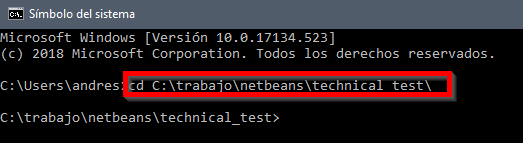
# Diagrama de casos de uso



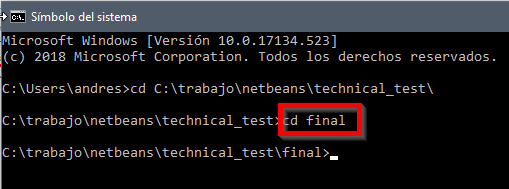
# Ejecucion

La API se ejecuta en un servidor Payara Micro como se muestra a continuación:

En un terminal ingresar a la carpeta del proyecto, en mi caso se encuentra en la ruta “C:\trabajo\netbeans\technical\_test”



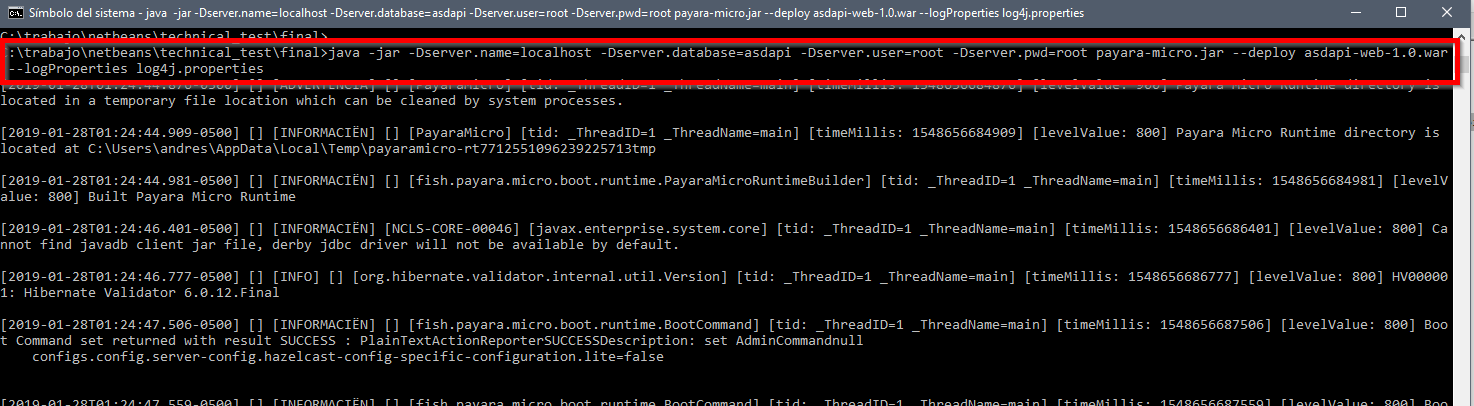
Luego ingresar a la carpeta final



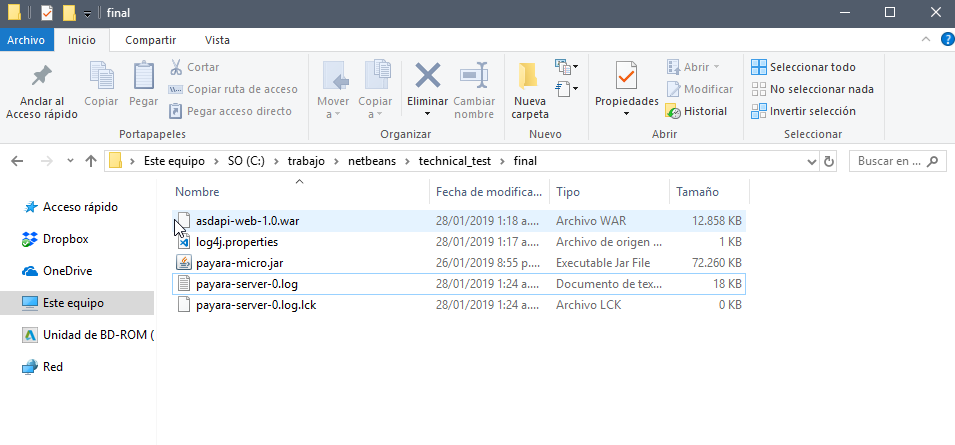
Y aquí ejecutar el siguente comando:

java -jar -Dserver.name=localhost -Dserver.database=asdapi -Dserver.user=root -Dserver.pwd=root payara-micro.jar --deploy asdapi-web-1.0.war --logProperties log4j.properties

Donde los parámetros “-Dserver.name=localhost -Dserver.database=asdapi -Dserver.user=root -Dserver.pwd=root” establecen la configuración de base de datos

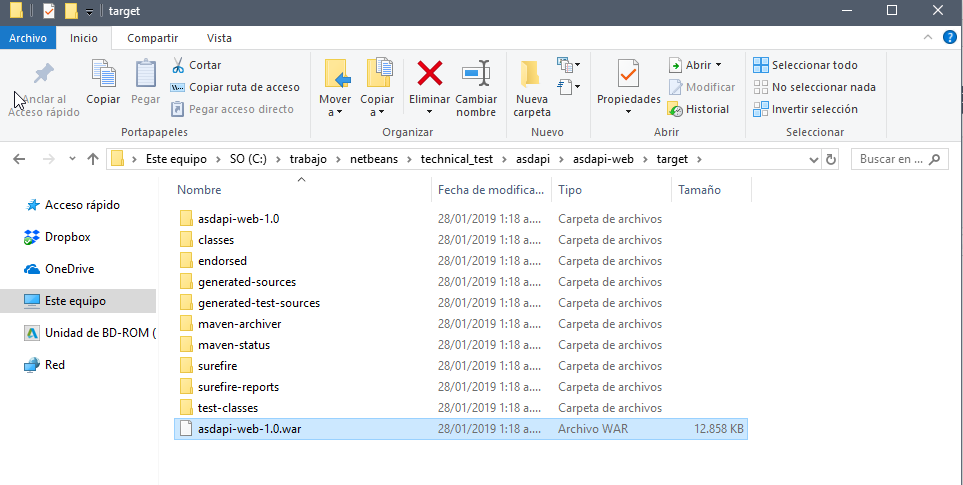


La carpeta final debe contener los siguientes archivos:



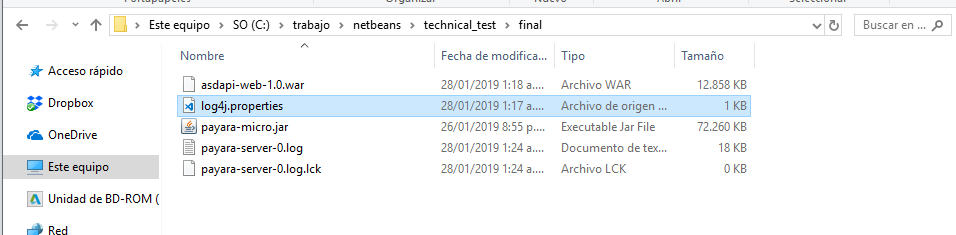
El archivo .lck se crea cuando el servidor inicia la ejecución

El archivo “asdapi-web-1.0.war” es el desplegable generado por el IDE se puede encontrar en la siguiente carpeta después de la compilacion:

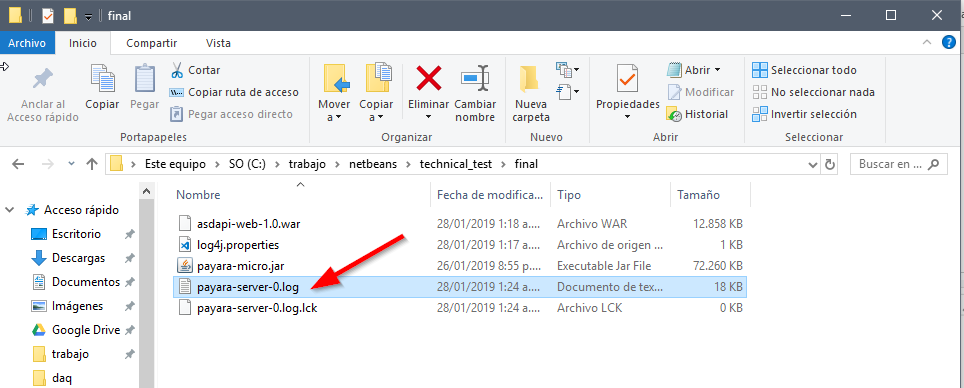


# Log4j

La configuración de log4j se encuentra en el archivo “log4j.properties”:



Archivo de log



# Configuración Junit

Debido a que junit debe ejecutar ejb y jpa fue necesario colocar en el proyecto maven un contenedor embebido de ejb`s lo único que se debe configurar para esto es la conexión con base de datos, si no se configura la conexión los casos de prueba fallaran en la compilación.

Para configurar la conexión se debe editar las siguientes propiedades del archivo domain.xml incluido en el proyecto

<property name="User" value="root"></property>

<property name="Password" value="root"></property>

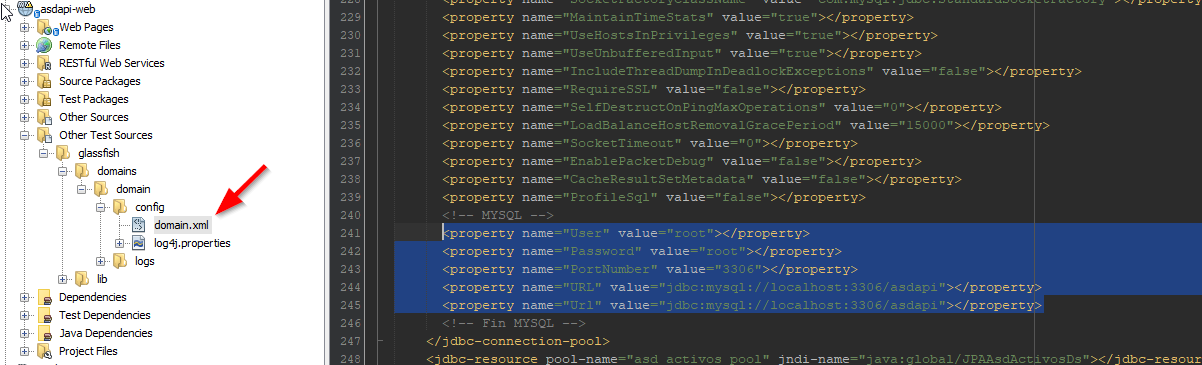
<property name="PortNumber" value="3306"></property>

<property name="URL" value="jdbc:mysql://localhost:3306/asd\_activos\_api"></property>

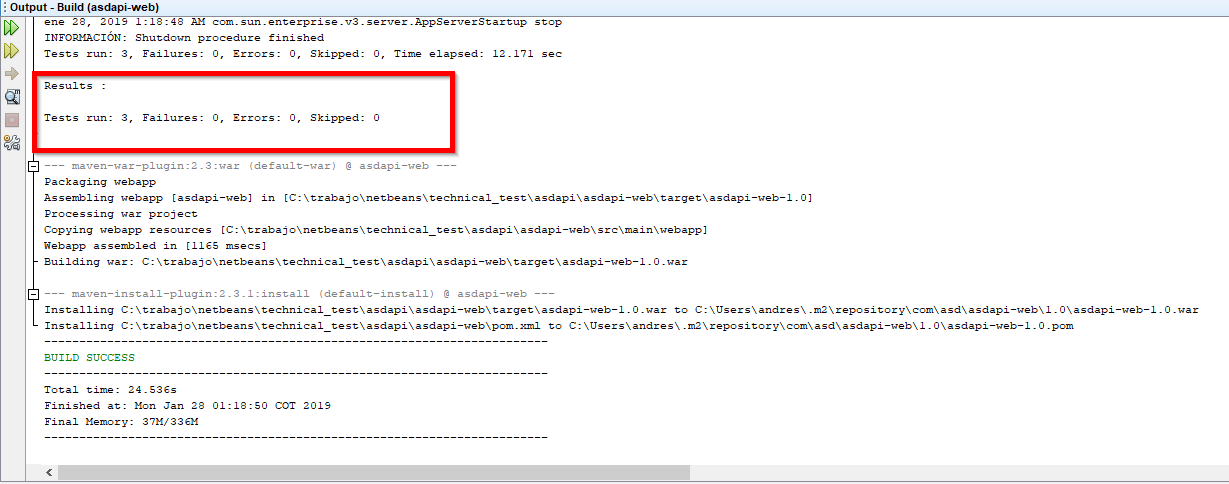
<property name="Url" value="jdbc:mysql://localhost:3306/asd\_activos\_api"></property>

Por favor prestar especial atención en la configuración de **URL** y **Url** ambas deben ser iguales

El archivo domain.xml se encuentra en la siguiente ubicación:



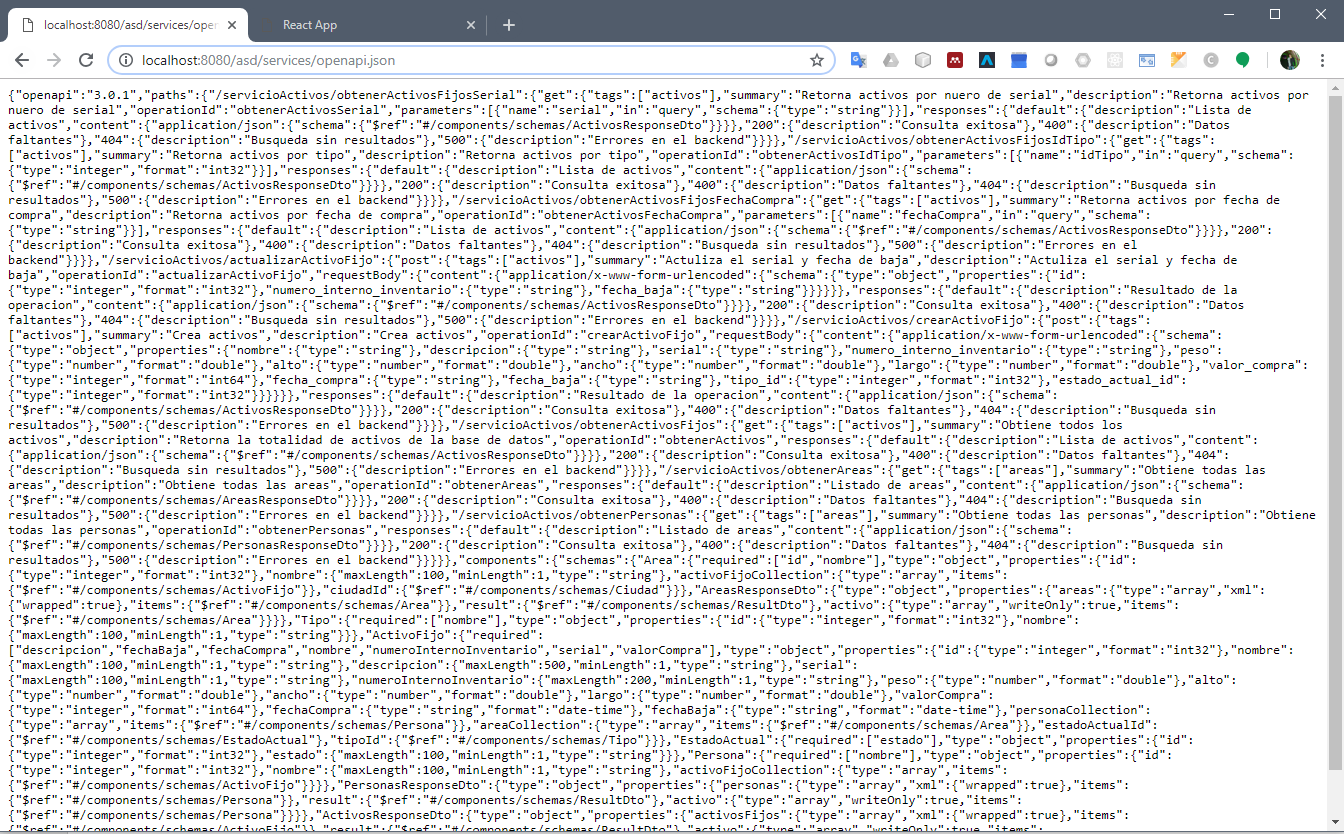
Resultado de la compilación:



# Documentación Swagger

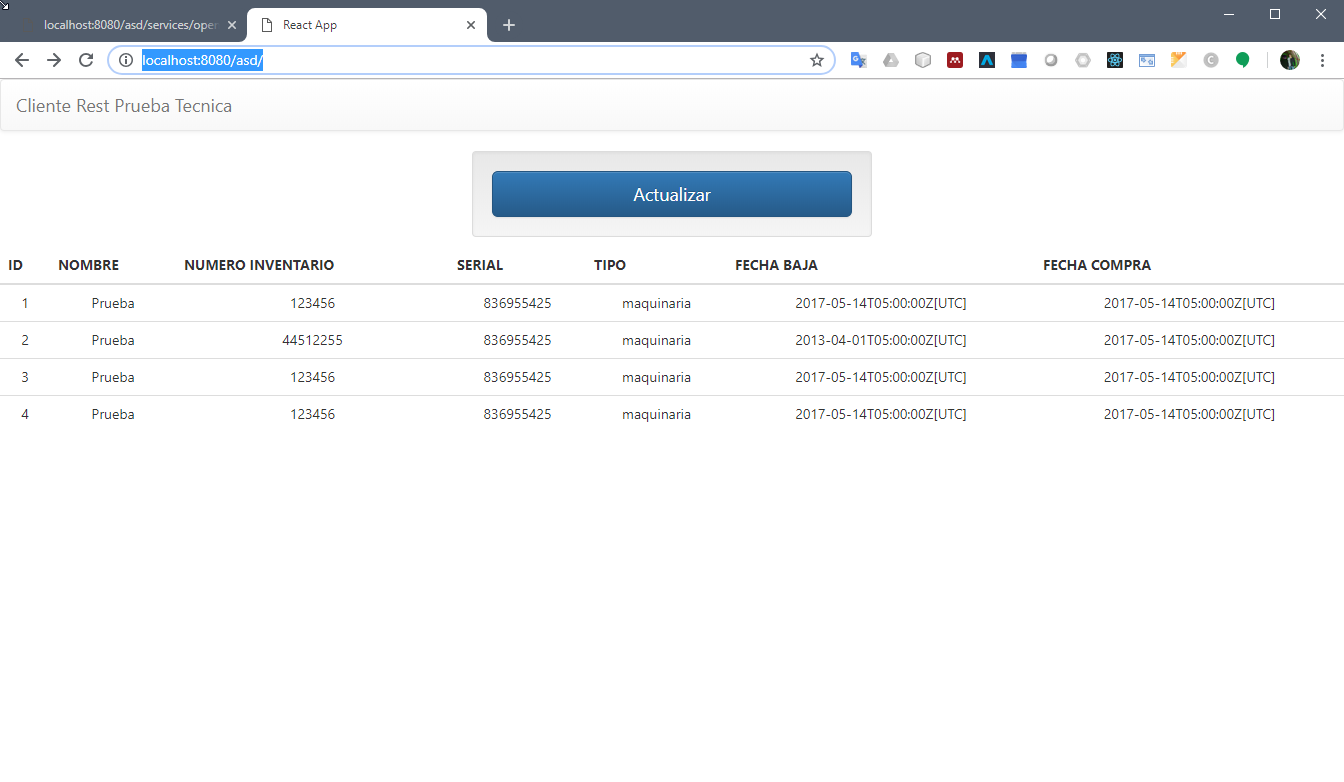
El servicio rest se encuentra documentado con swagger para ver la documentación se puede ingresar a la URL:

<http://localhost:8080/asd/services/openapi.json>



# Cliente servicio web

Se creo un cliente sencillo en ReactJs el cual consume el sevicio “obtenerActivosFijos” a este cliente se puede acceder mediante la URL: <http://localhost:8080/asd/>

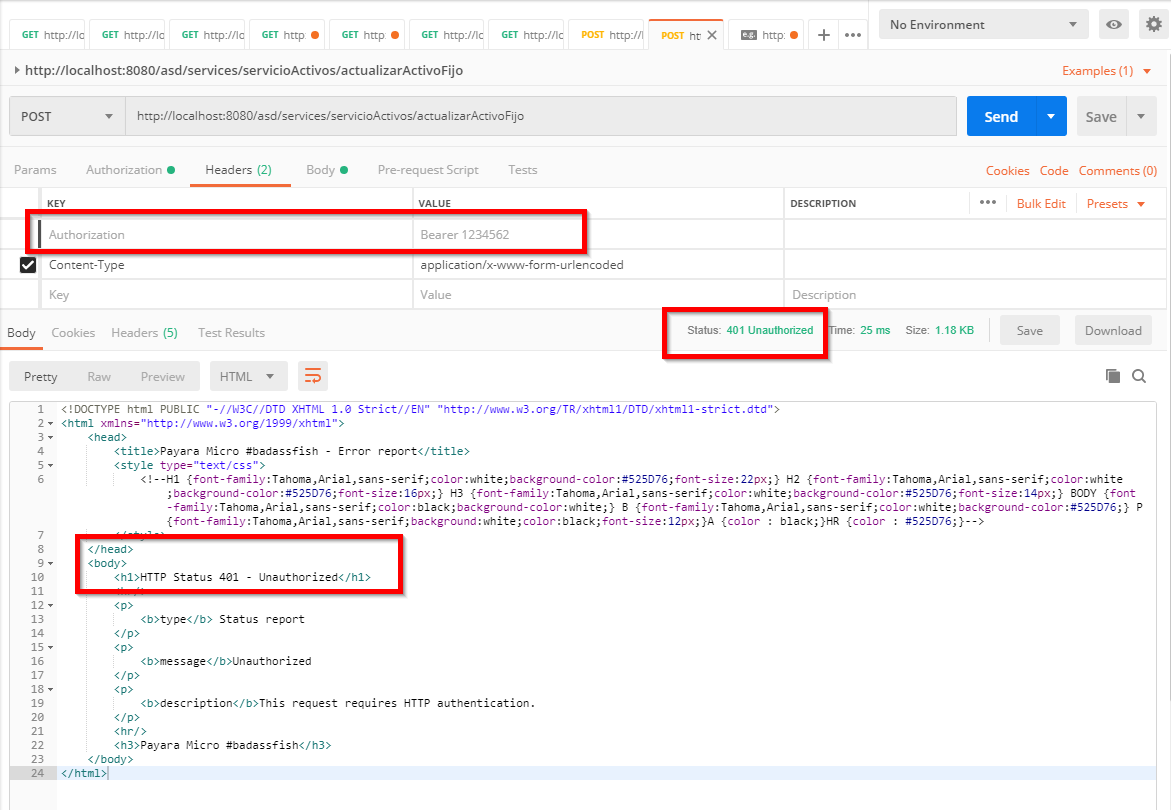


Los códigos fuente del cliente se encuentran en la carpeta “technical\_test\reactclient”

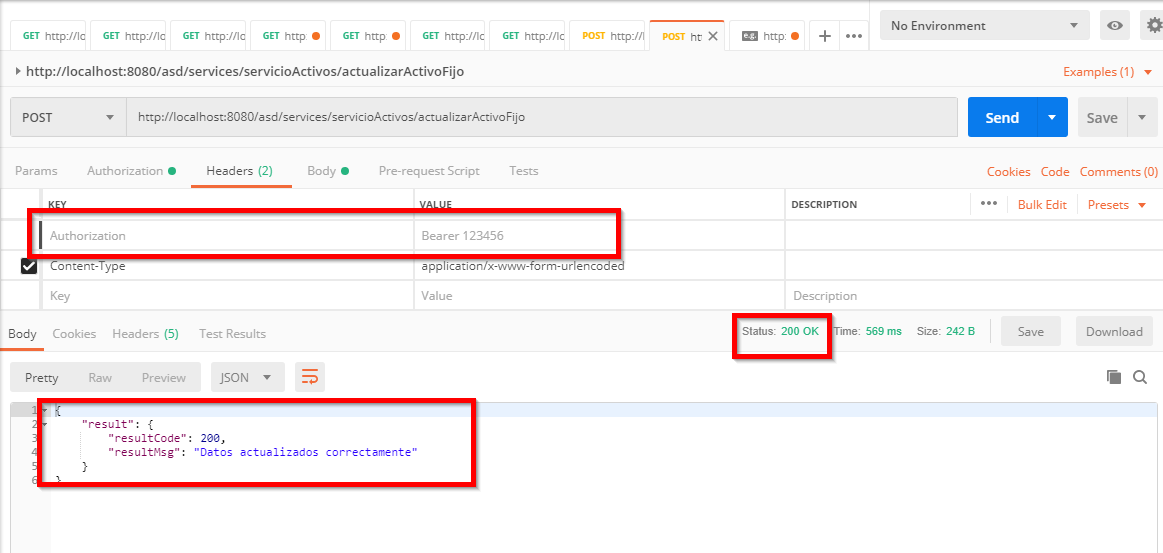
# Autenticación por token

Para los servicios “obtenerPersonas” y “actualizarActivoFijo” se programo un sistema de autenticación por token, el token esta fijado por código y este es “123456” el token debe ser enviado en la cabecera del servicio “Authorization: Bearer 123456”

Cuando se envía un token invalido a cualquiera de estas dos funciones se obtendrá la siguiente respuesta:



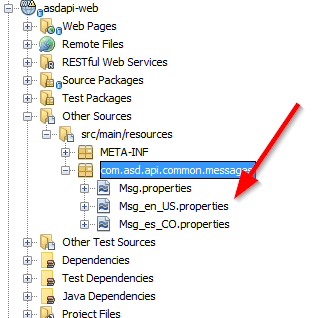
Cuando el token es valido el servicio se ejecutará correctamente:



# Internacionalizacion i18n

El servicio cuanta con internacionalización para los mensajes de respuesta

Los textos de los mensajes se encuentran en los siguientes archivos:



# GIT

Todo el desarrollo de la aplicación se encuentra versionado en Git

