**Nombre: Ana María Álvarez Carreño.**

Prueba Técnica Proceso Software Engineer**.**

**Paso 1:**

Se presenta un script en donde se debe crear una base de datos llamada ***cyxtera***, se debe crear un schema llamado operaciones y ejecutar los comandos, ver archivo adjunto ***bd\_cyxtera***.

-- SCHEMA: operaciones

-- DROP SCHEMA operaciones ;

CREATE SCHEMA operaciones

AUTHORIZATION postgres;

COMMENT ON SCHEMA operaciones

IS 'Auditoria de operaciones realizadas';

-- Table: operaciones."Auditoria"

-- DROP TABLE operaciones."Auditoria";

CREATE TABLE operaciones."Auditoria"

(

"idAuditoria" serial NOT NULL,

"Usuario" character(200) COLLATE pg\_catalog."default" NOT NULL,

"Transaccion" character(1) COLLATE pg\_catalog."default" NOT NULL,

"FechaCreacion" date NOT NULL,

"Servicio" character(200) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "pkAuditoria" PRIMARY KEY ("idAuditoria")

)

WITH (

OIDS = FALSE

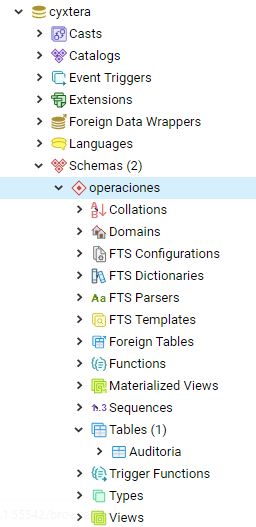
)

TABLESPACE pg\_default;

ALTER TABLE operaciones."Auditoria"

OWNER to postgres;

**Despliegue de bases de datos**



**Paso 2:**

El servidor que se utilizó para este desarrollo fue wildfly-15.0.1. Final, en donde se tiene que comprobar que esté instalado el driver de PostgreSql, configurando el standalo.xml en copiando las siguientes líneas:

<datasource jndi-name="java:jboss/env/jdbc/cyxtera" pool-name="cyxtera">

<connection-url>jdbc:postgresql://localhost:5432/cyxtera</connection-url>

<driver-class>org.postgresql.Driver</driver-class>

<driver>postgresql</driver>

<security>

<user-name>postgres</user-name>

<password>XXXX</password>

</security>

</datasource>

<driver name="postgresql" module="org.postgresql">

<driver-class>org.postgresql.Driver</driver-class>

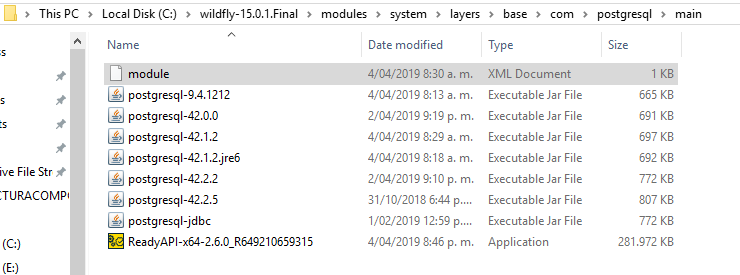
<xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>

</driver>

Esto con el fin de realizar la conexión a la base de datos la cual fue realizada en PostgreSql.

Ubicación de driver, en este caso se está utilizando la versión postgresql-42.1.2, para el driver de conexión.

**Módulo de Servidor conector PostgreSql**

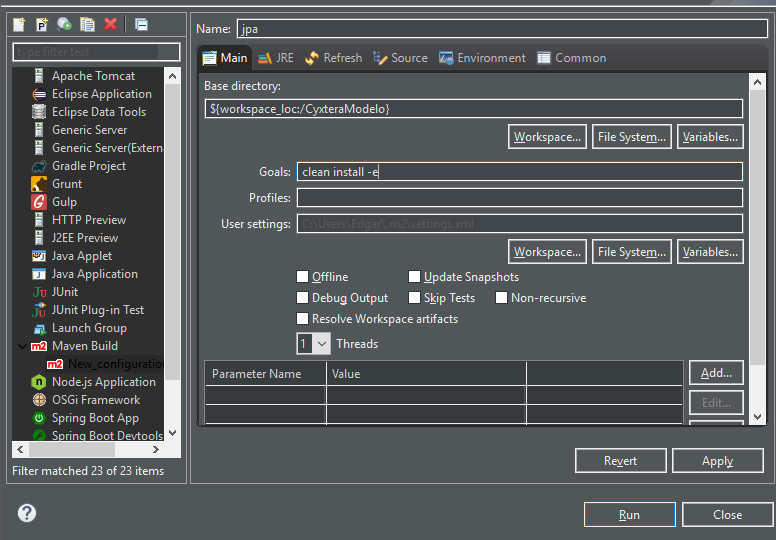


**Paso 2:**

Para realizar el despliegue se deben generar las siguientes tareas maven:

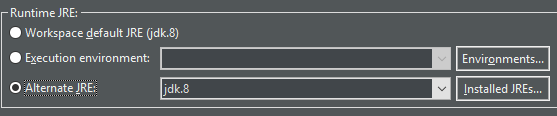
Crear por cada proyecto un maven Build y colocar en Goals las siguientes instrucciones: clean intall –e, en cada uno de los proyectos presentados:

**Tareas maven a ejecutar para despliegue**

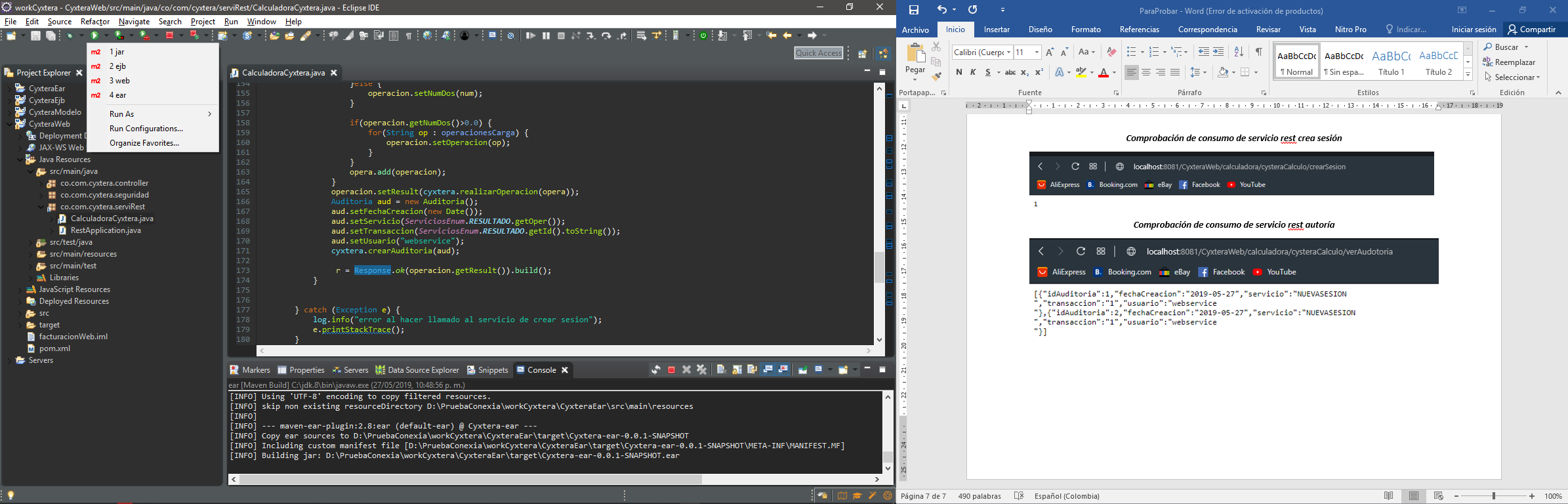


Se recuerda que se debe instalar la jdk 1.8 para que la compilación sea correcta.

**Configuración de jdk 1.8 en tareas maven.**

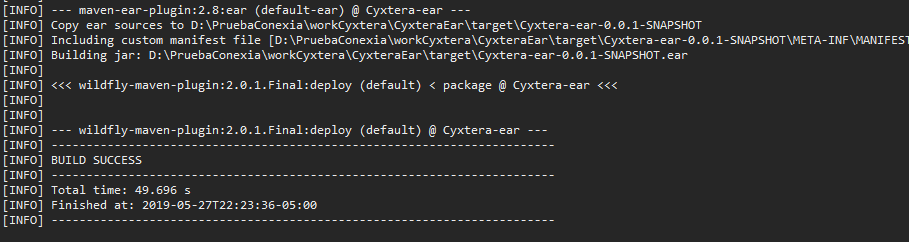


**Orden de ejecución de las tareas maven en el orden que se presenta**



Se comienza a compilar siempre por el proyecto modelo o jar, después por el ejb, web y cerrando por el ear para que realice el despliegue directo al servidor.

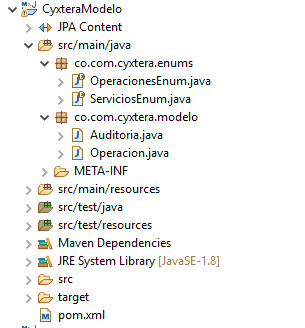
***Comprobación de que el proyecto se ha compilado correctamente***



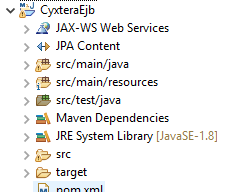
**Paso 3:**

El proyecto se divide en los siguientes paquetes, se hace referencia que son proyectos tipo maven:

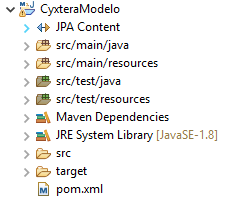
CyxteraModelo: en donde se encuentra el mapeo de las entidades a la base de datos, las enumeraciones propias que hacen alusión a las constantes que se utilizaran, el objeto que hará control de las operaciones y los números generados.



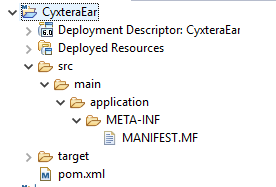
CyextetaEjb: proyecto encargado de tener la lógica del negocio, el cual se expone por medio de una fachada tipo remoto, un Bean de la persistencia y lógica y por ultimo otro Bean de no transaccional para como transporte de datos de capa segura.



CysteraWeb; proyecto encargado de crear los servicios Rest que se van a consumir y exponer, se implementa una pantalla tipo web como prueba de servicio.



CyxteraEar: proyecto que empaqueta los proyectos anteriormente mencionados y el que se despliega en servidor.



Se deba mencionar que el servidor debe estar encendido al momento de realizar el despliegue.

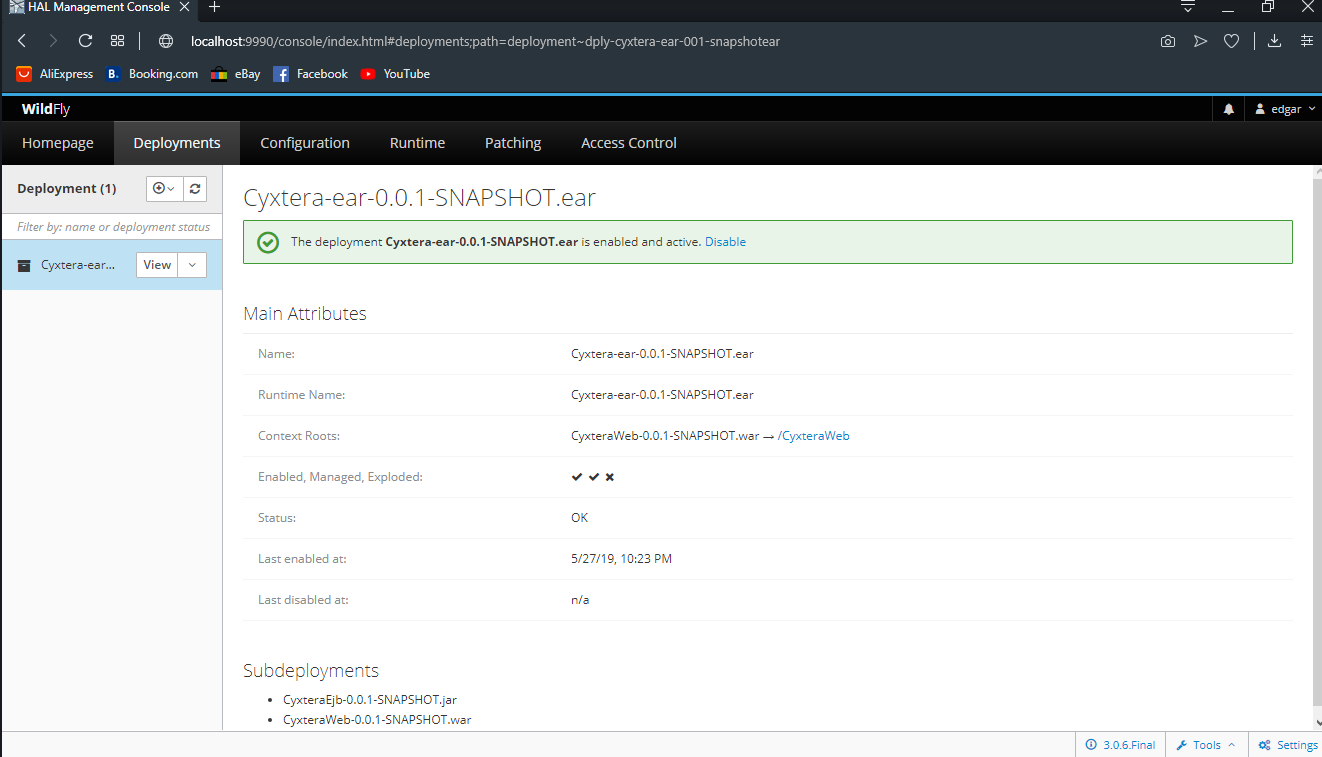


El proyecto está configurado para que por medio de las tareas maven este realice el respectivo despliegue:

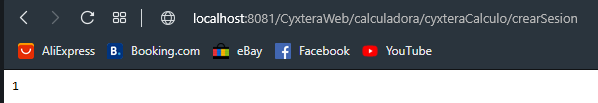
**Comprobación de despliegue de proyecto en servidor**



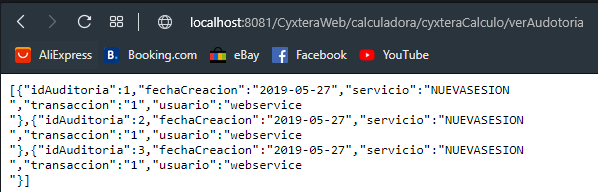
***Verificación en entorno gráfico del servidor:***



***Comprobación de consumo de servicio rest crea sesión***

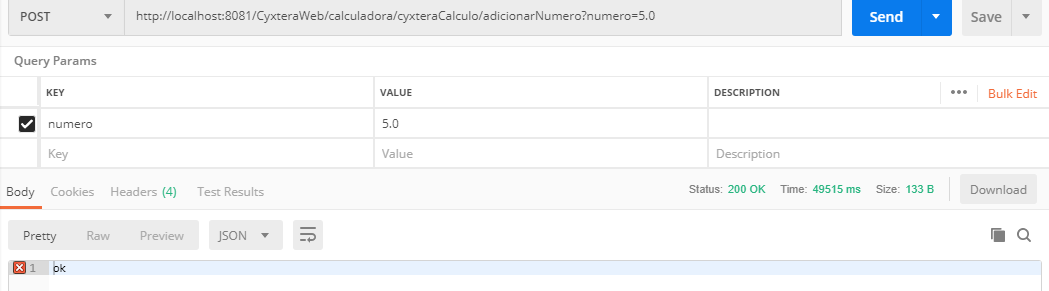


***Comprobación de consumo de servicio rest autoría***

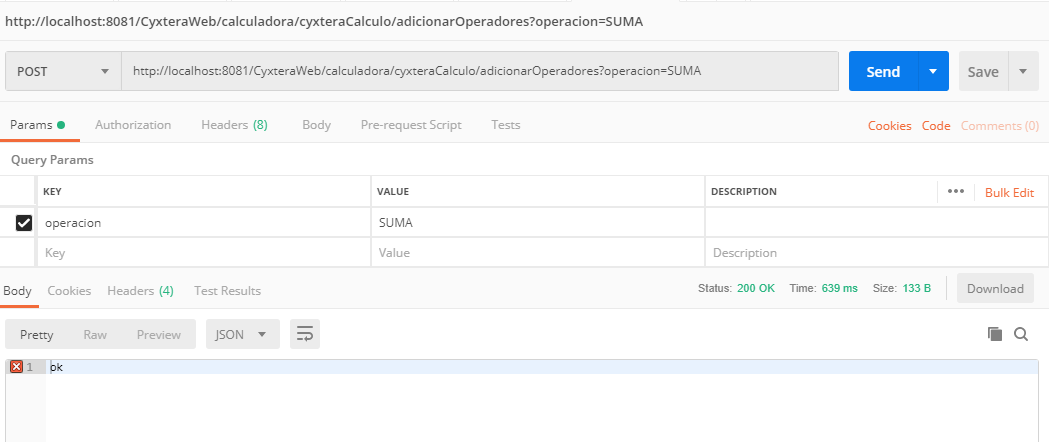


***Los servicios rest que tiene definición Post fueron probados con el asistente postman***

***SERVICIO ADICIONAR NUMERO***

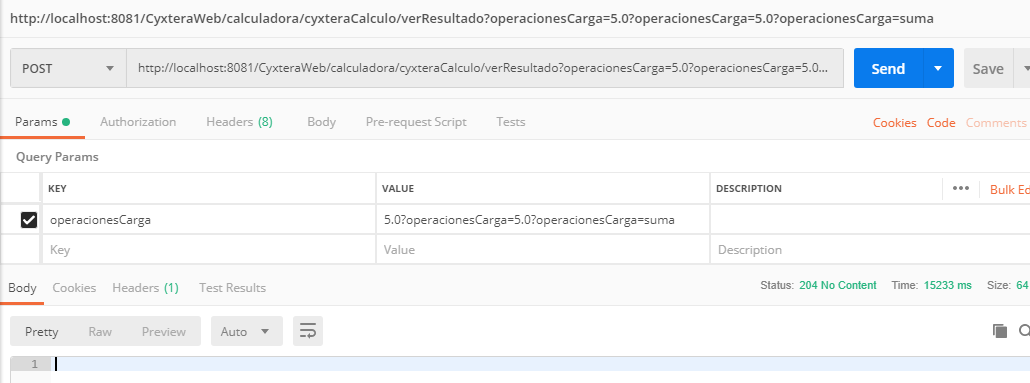


***SERVICIO ADICIONAR OPERACIONES***



Por cada ejecución de un servicio se es realizado la persistencia en base de datos

***SERVICIO OPERAR***



Por cada ejecución de un servicio se es realizado la persistencia en base de datos

Con estos servicios funcionando se realiza la comprobación por medio del entorno gráfico con la siguiente pantalla:

