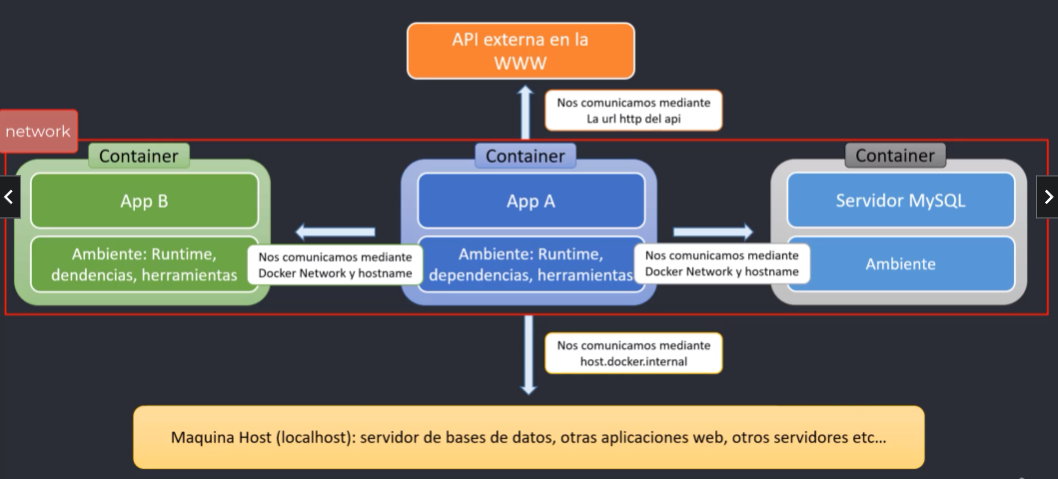
8.  Docker Networks: Comunicación entre contenedores

Comunicaciones en Docker y volúmenes es donde quedan los datos persistidos porque la maquina se reinicia y sus variables de ambiente para utilizarla en los contenedores

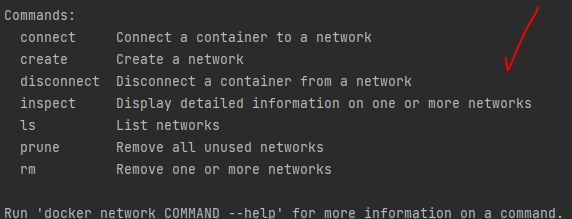


Dockerizando microservicio cursos y configurando la red o network

Procedo a dockerizar el microservicio Curso

* Configurando el datasource para que apunte a la db desde el container (host.docker.internal)
* Ajustando el la url del feign (Puede ser solo el nombre del proyecto que esta configurado en app.properties)
* Como no manejamos la estruturade los proyecto ejecuto el Jar por independiente NO OPTIMIZADO
* Configuro el archivo dockerfile (Lo pego del otro proyecto)
* Creando la imagen de curso

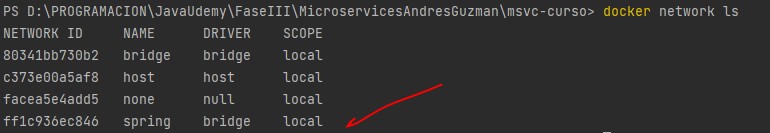
Luego procedo ver los comandos network para enalazar los container del proyecto docker network --help



Procedo a crear la red network con el nombre de spring



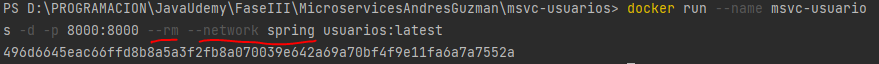
Listando las redes



 Comunicación entre contenedores

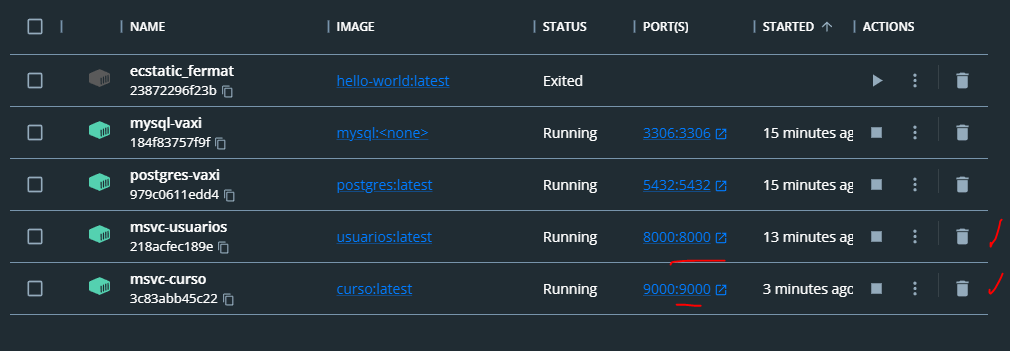
Elimino los contenedores de usuario y ya actualizadas las imágenes de curso y usuarios procedo a

* crear los contendedores usuario y curso, eliminarlos en cuanto se dejen de utilizar

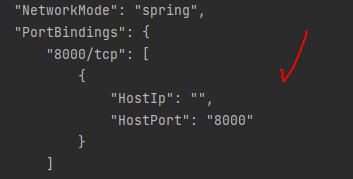
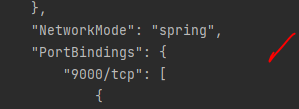




* Corriendo los container procedo a ejecutar pruebas y todo esta OK



* Procedo a validar con inspect los containers y corroborar que el network sea spring (La ip no se recomienda utilizar ya que es variable)



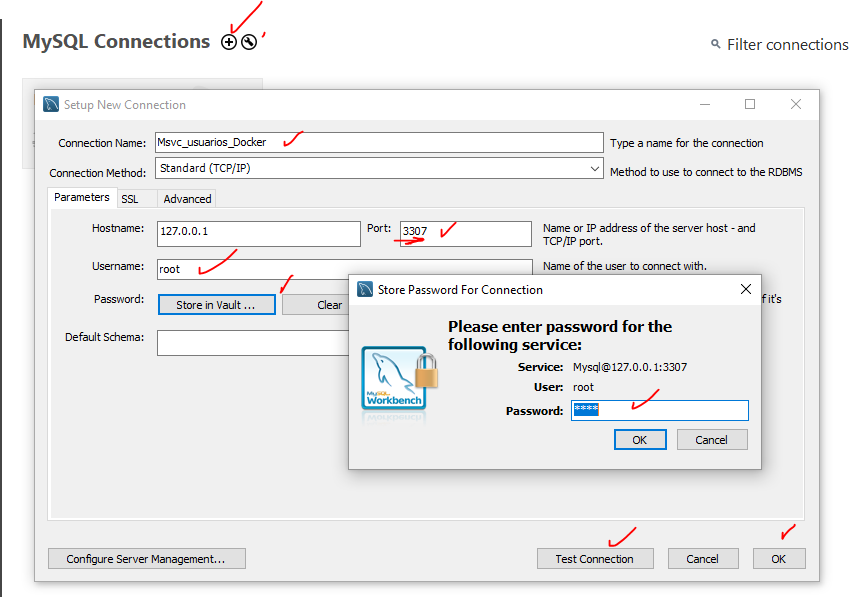
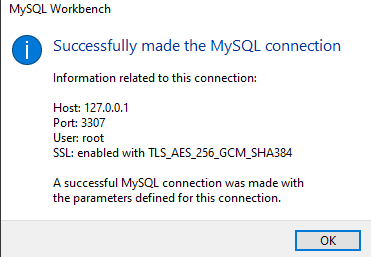
Dockerizando MySQL

Descaro la imagen de dockerHub(Ya la tenia) y realizo el pull para luego crear el contenedor de mysql

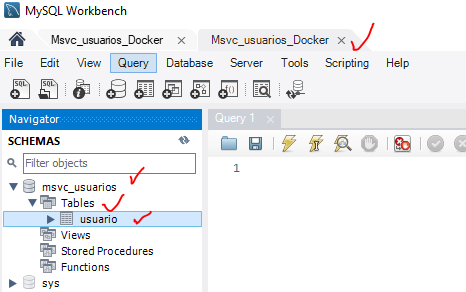
docker run --name mysql-aguzman -d -p 3307:3306 --network spring -e MYSQL\_ROOT\_PASSWORD=sasa -e MYSQL\_DATABASE=msvc\_usuarios mysql:latest

OJO: Cuando este creando un container a nivel de DB tratar de evitar que los puertos externos usen localmente los que traen la DB por DEFAUTL ejemplo si es mysql quitar el 3306 posgtre o cualquier DB. importante para que no presente errores de conexión a motores. importante agregar la DBs(containers) al network (Red de trabajo = spring), para este dockerizazion la DB no tendrá esquema, se empleara el esquema publico.

1. Nueva Conexión
2. Nombre de la conexión, ideal referente al microservicio que se implementa
3. Puerto de escucha en este caso configurado en Docker fue 3307
4. Usuario
5. Password
6. Validación de conexión
7. Realizar conexión

Procedo a revisar si el ambiente me creo la tabla usuario o realizo un refresh



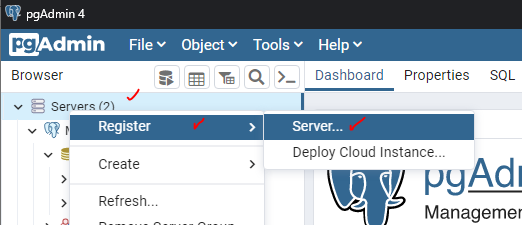
 Dockerizando PostgreSQL

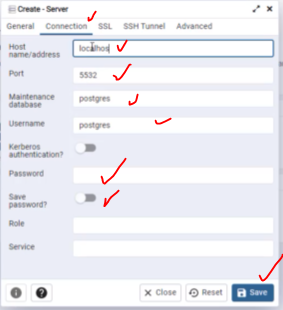
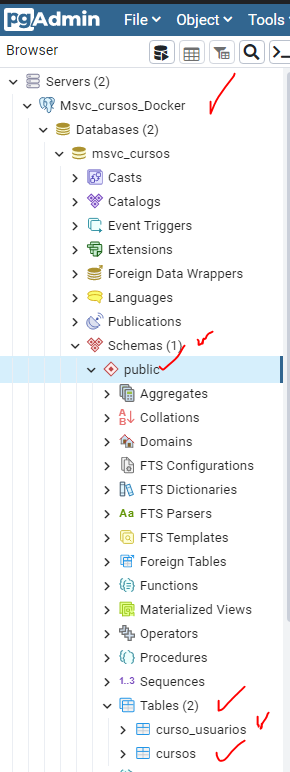
Aprovecho para descargarlo localmente en <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>. Instalación -> Todo siguiente, agrego clave y usuario postgres, instalar(quitar paloma de instalar stack y terminar).

Creo el nuevo container de DB postgres, para este dockerizazion la DB no tendrá esquema, se empleara el esquema publico

docker run --name postgre-aguzman -d -p 5532:5432 --network spring -e POSTGRES\_PASSWORD=sasa -e POSTGRES\_DB=msvc\_cursos postgres:latest

Agregando la interfaz del container postgre a el motor Postgre, Procedo a revisar si el ambiente me creo la tabla usuario o realizo un refresh.



Comunicación contenedores con BBDD Dockerizadas (MySQL y PostgreSQL)

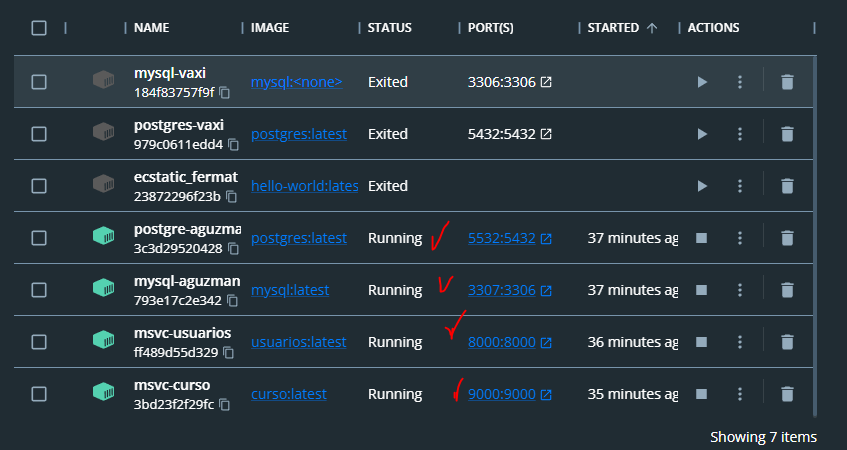
Ajusto los datasource el host que tendrá sera el nombre del container de cada DB





Se puede realizar así por tema de la RED de trabajo NEKWWORT

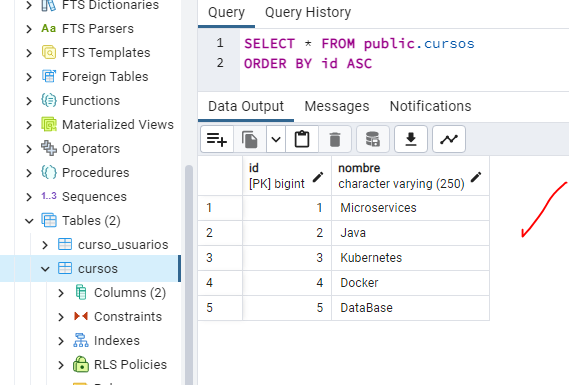
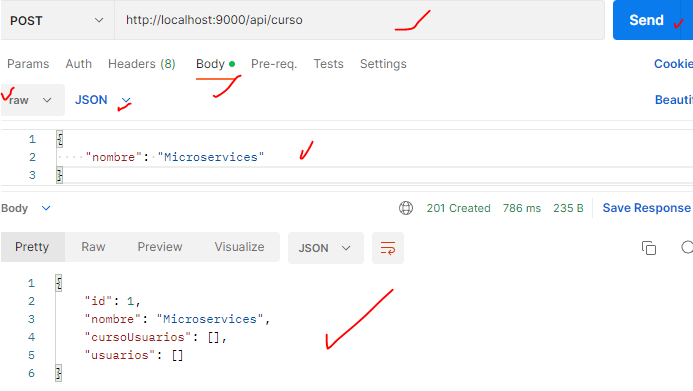
* Compilar nuevamente los Jar, crear la imagen del SERVICIOS y crear nuevamente los contenedores.
  + ./mvnw.cmd clean package -DskipTests
  + docker build -t curso .
  + docker run --name msvc-curso -d -p 9000:9000 --rm --network spring curso:latest
  + ./mvnw.cmd clean package -DskipTests
  + docker run --name msvc-usuarios -d -p 8000:8000 --rm --network spring usuarios:latest
  + docker build -t usuarios .



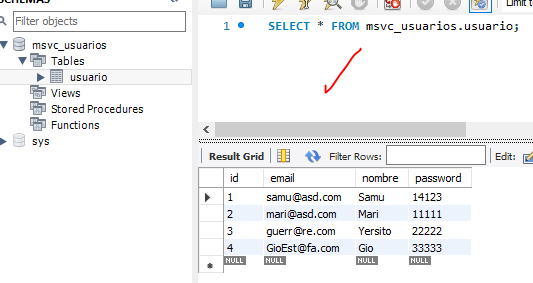
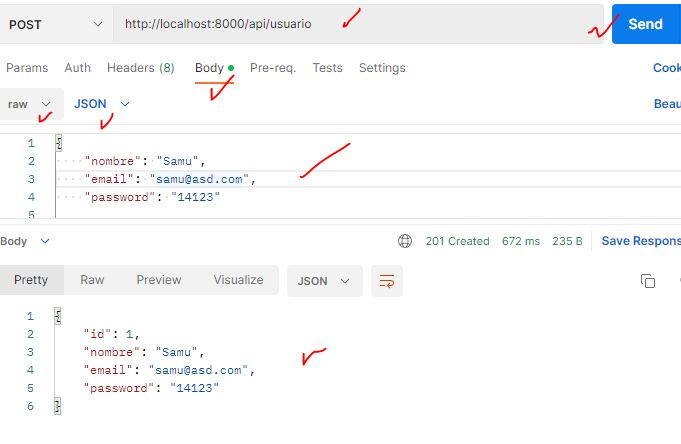
Revisando microservicios dockerizados en Postman

Realiando Pruebas de comunicación y agregando nueva data

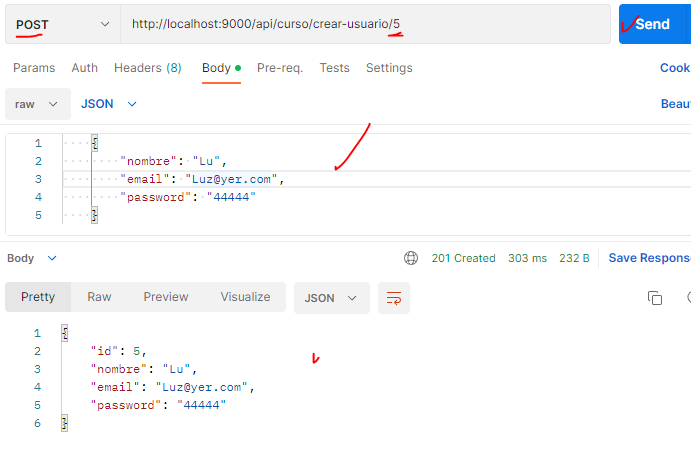
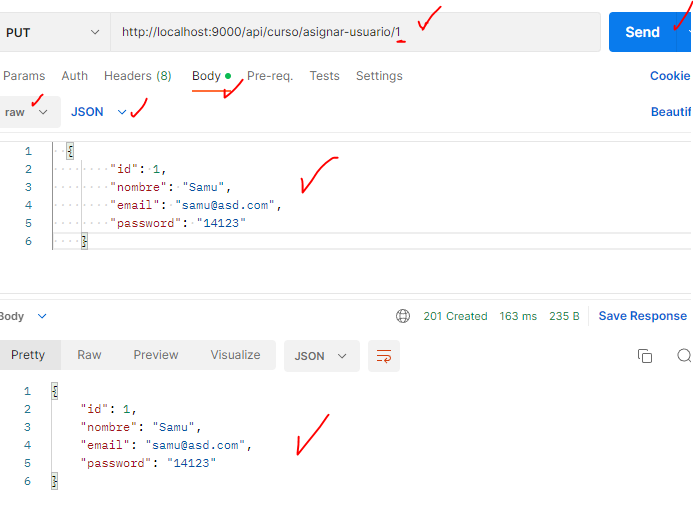
* Agregando cursos, microservicios, kubernetes, db, Docker y java y valido en Posgret(Docker)

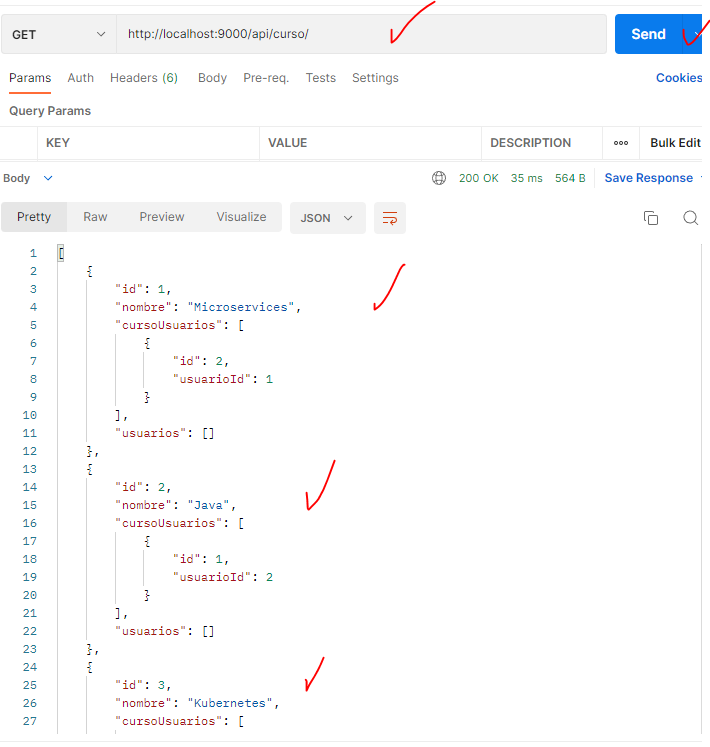
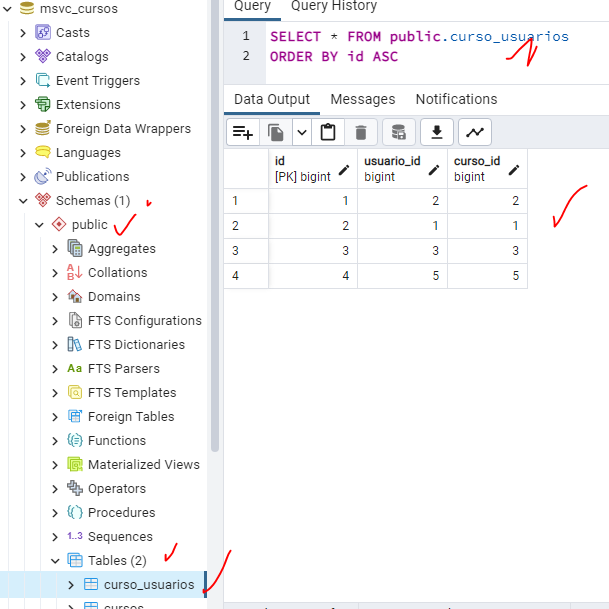


* Agregando nuevos usuarios, Samu, Mari, Yersito, Gio y corroboro en Db Mysql(Docker)



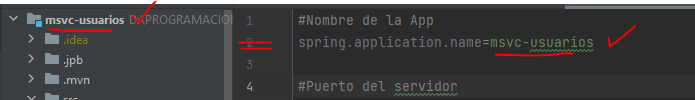
* Asignar curso a un usuario y creando Usuario en un curso(id=5) especifico y validar los usuarios cpn curso en DB





Como vemos todas las pruebas se han realizado con 100% de comunicación entre los dos servicios

OJO: Importante como se esta manejando la ocmunicacion de los servicios por nombre el nombre del app.properties debe ser igual al nombre del proyecto y no puede cambiar si no la comunicación de los proyectos por name dará error.



Problema con persistencia de datos en MySQL/Postgres al eliminar el contenedor

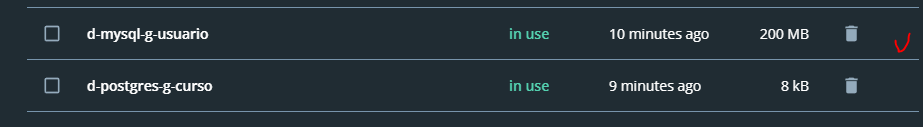
¿Que pasa si elimino los contenedores de DB? 🡪 Efectivamente la data se va borrara así volvamos a ejecutar el mismo comando para la generación de los contenedores DB o ejecutando el comando para refrescar los contenedores DB Docker run --restart=always. La solución a esto en la siguiente Clase

Docker Volumes: la solución al problema de persistencia de datos

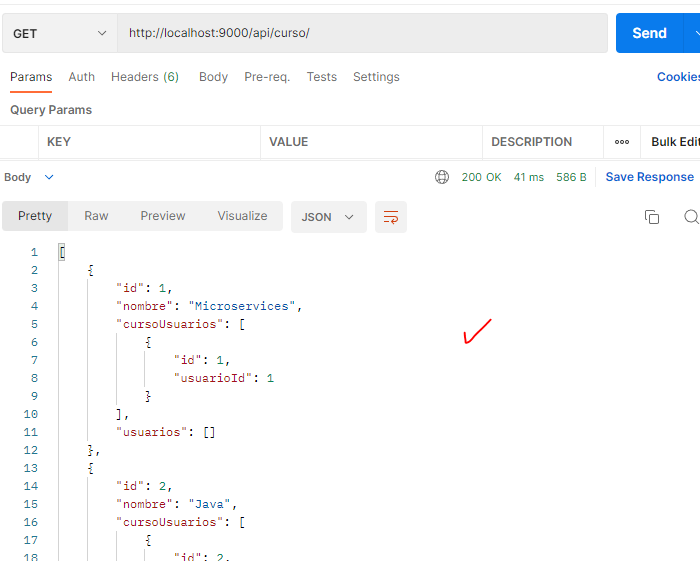
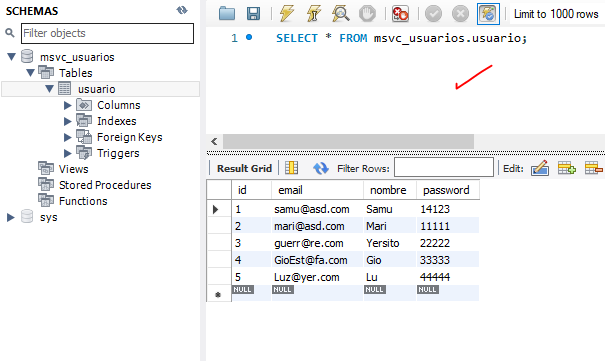
Solucionando el problema de la data persistida en la DB

* Detener y eliminar los contenedores de DB y Microservicios
* Crear contendedores de DB con Volumen(-v) asignada la ruta según el motor de DB y le damos la condición que se reinicie cada vez que se detenga o ocurra algo en automático(Docker run --restart=always)
  + docker run --name mysql-aguzman -d -p 3307:3306 --network spring -e MYSQL\_ROOT\_PASSWORD=sasa -e MYSQL\_DATABASE=msvc\_usuarios -v d-mysql-g-usuario:/var/lib/mysql --restart=always mysql:latest
  + docker run --name postgre-aguzman -d -p 5532:5432 --network spring -e POSTGRES\_PASSWORD=sasa -e POSTGRES\_DB=msvc\_cursos -v d-postgres-g-curso:/var/lib/postgres --restart=always postgres:latest

OJO: Importante cuando cree el volumen en el local debo apuntar la ruta correcta para cada DB si no cunado realice el procedimiento la data no se alamcenara.



* Crear los contenedores de los microservicios y levantar
* Prueba, Agregando la data y probando que todos los métodos corran OK
* Prueba validando si se guarda la data en lops volúmenes
  + Detener containers,
  + eliminar los containers en especial los de DB
  + Validar si existen los volumen
  + Consultar en DB si existe la data (NO EXISTE)
  + Crea nuevamente los container DB y levantar los container de servicio para validar si la data aún existe (SI EXISTE).

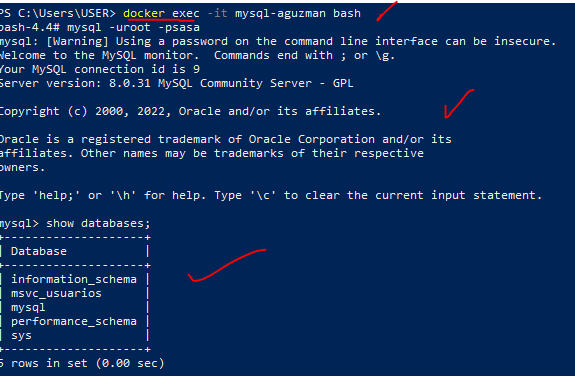
 Conectarse desde contenedor cliente de línea de comandos a MySQL/Postgres

Este contenedor es utilitario permite ejecutar comandos sql es como un cliente(No es una instancia) contra la DB. Yo lo conozco como entrar al bash o Shell de un DB dockerizada para el ejemplo sera la de mysql.

* Ingresando al bash (Necesariamente todos los container levantados, en especial al que validaremos en este caso mysq-gusman)

Existen varias formas de entrar por medio de la imagen del contenedor o un exec sobre el container a validar

1. Docker exec (Ya se puede administrar toda la DB) 🡪 apunta al contenedor



1. Docker run (Ya se puede administrar toda la DB) 🡪Crear un contenedor utilitario. Apunta a la imagen DB(mysql:latest) y luego al contenedor mysql-aguzman , lo ideal cuando se use esta forma es que en el comando se agrege un --rm para que al finalizar el uso de este se elimine

