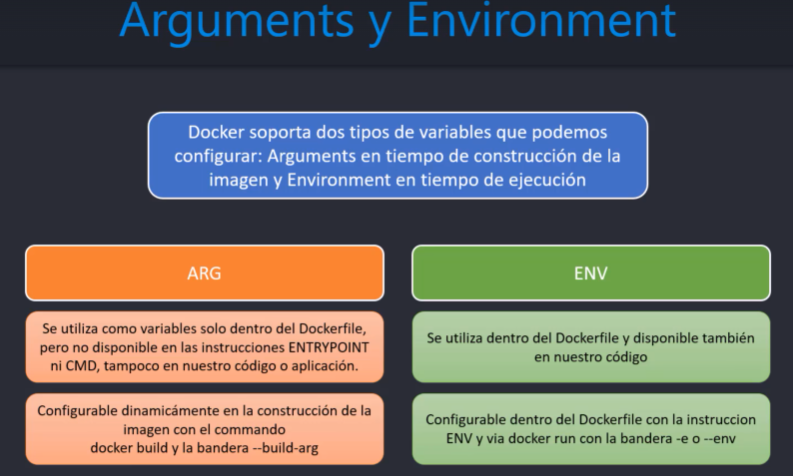
9. Docker: Arguments y Environment Variables

Introducción

Arguments 🡪 Para la contruccion de la imagen

Enviroment 🡪 En tiempo de ejecución

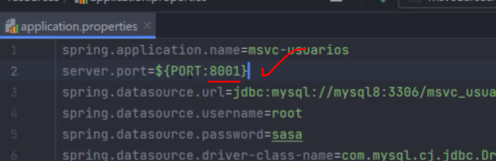
Facilita la configuracion



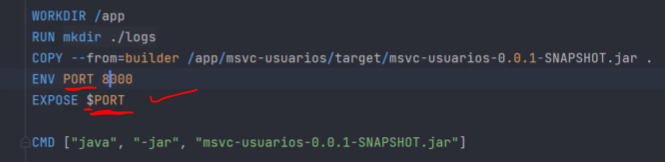
Trabajando con variables de ambiente

Para este caso el puerto del microservicio será dinámico por medio de una variable de el puerto en el properties es 8081 pero el del dockerfile 8080, como el puerto es dinámico tomara como principal el que le asigne el dockerfile

* Configurando el properties



* Configurando el Dockerfile que será el que otorgue el verdadero puerto interno al proyecto, trata de que el puerto dinámico quede en la ultima capa de ese modo no detiene tanto si se debe recompilar.



* Creando la imagen del proyecto con esa variable de ambiente
* Crear y levantar el contenedor del proyecto a puntando al puerto del dockerfile

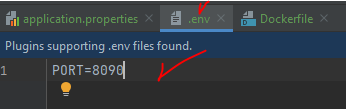


* Se probo y funciona bien entonces como es un puerto interno dinámico procedo a cambiarle el valor al puerto y este de ejecución(--env) sobrescribirá el puerto interno del dockerfile



* Otra forma de ejecutar variables de envairoment
* Crear archivo envariroment(.env) y ejecutar el contenedor llamando al archivo .env y agrego la ruta

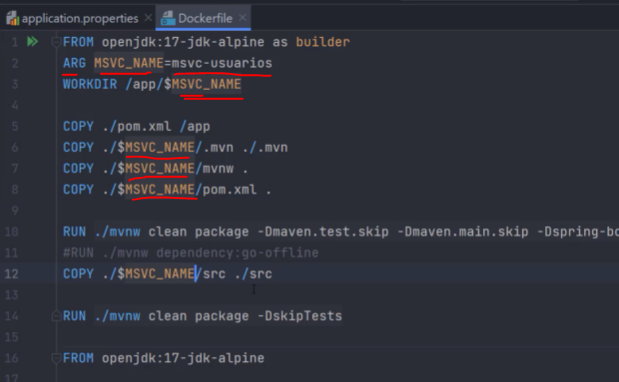




 Trabajando con argumentos en el Dockerfile

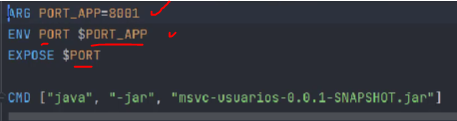
Para este caso el nombre del proyecto se utiliza mucho en este caso la convertiremos en una variable por si en un futuro llega a cambiar la variable se de modo genérico y practico el cambio.

* Defino la variable con ARG de argumento



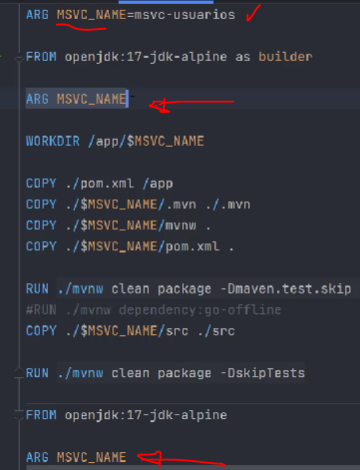
Ojo: la variable funciona para comando que se van a ejecutar sobre la imagen no mas, como el nombre del jar o la ruta que contiene el jar

* Crear imagen y crear el Docker del proyecto para validar
* Opcional asginar el puerto interno desde la imagen



Y se crear la imagen con el puerto 8081 en el dockerfile pero desde la ejecución se le asigna el puerto 8080

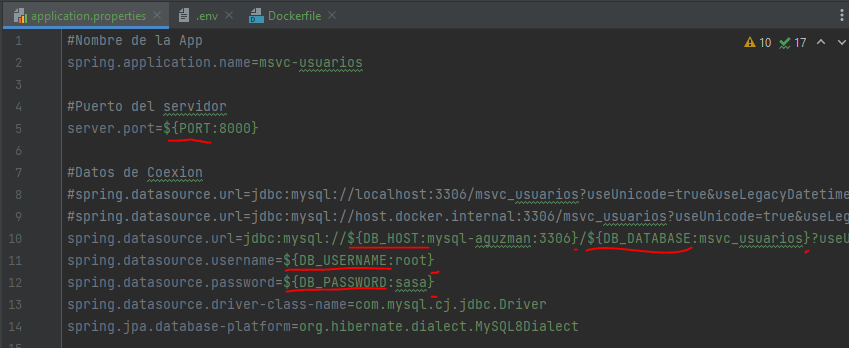
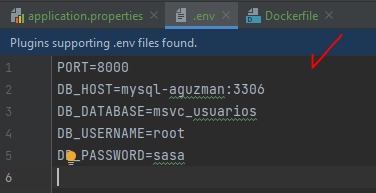
Si nuestros dockerfile presenta varias capas declaro la variable en la primera línea y luego realizo el llamado después del from para utilizar en esa capa



Variables de ambiente (env) para los parametros de MySQL y Postgres

Los cambios se realizaran a nivel de Datasource para postgrest y mysql.

Para este caso se manejaran las variables enviroment del app.properties donde se devife la variable y se le da el valor por default, pero el que tiene el poder de agregar los valores perfectos es el archivo .env

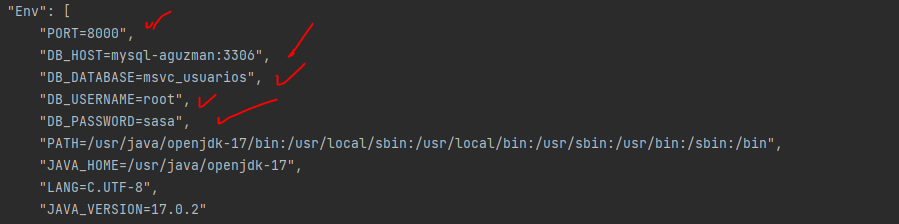
* Nuevamente generar el jar, la imagen y el contenedor de los proyectos

Revisando variables de ambiente DB con inspect y Postman

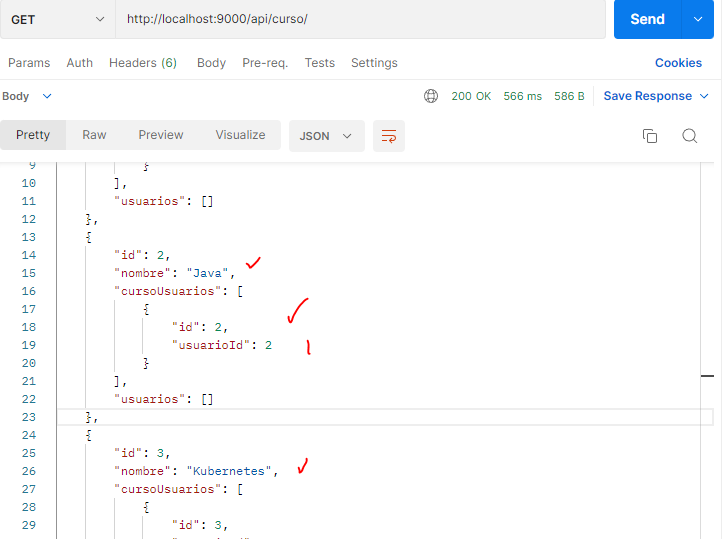
* Creando y levantando los contenedores de los microservicios
* Levantando los contenedores de DB
* Inspeccionó los contenedores de los microservicios para validar sus variables enviroment

docker run --env-file .env --name msvc-usuarios -d -p 8000:8000 --rm --network spring usuarios:latest





* Probando con Postman todos los cursos y los usuarios que estan en ellos



En conclusión es bueno tener las variables dinámicas ya que los datasource se puede realizar de modo ágil y que muy bien presentado.

Variables de ambiente (env) para los hostnames de los contenedores

La idea es realziar la conexión (URLs) al feign de modo dinámico(Solo con msvc-usuario)

* Creando las variables app.properties
* Creando la variable en el archivo .env
* Agregar la variable del properties en la clase respectiva feign
* Generar JAR, crear imagen y levantar servicio dockerizado
* Pruebas

