GitHub workflows:

Iedere keer als de code gepusht wordt, loopt hij de workflow af en updatet hij daarbij de image die je op DockerHub kan terugvinden.

Vier verschillende services met hun eigen Dockerfile. Dit kan misschien wel beter. Ik kreeg allerlei meldingen dat ik geen \*/jar file had en moest dan Maven installeren. Nadien moest ik nog een JAVA\_HOME path variabele aanmaken ook. Dus misschien dat hij daar bij jullie ook op vast loopt, maar ik vermoed dat jullie dezelfde versies erop hebben staan, behalve dan misschien de systeemvariabelen.

Ik heb:

JAVA\_HOME

C:\Users\matth\.jdks\corretto-17.0.10 (dus aan je eigen pad aan te passen)

MAVEN\_HOME

C:\Maven

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

Path

C:\Maven\bin

C:\Users\matth\.jdks\corretto-17.0.10\bin

Afbeelding met tekst, schermopname, software, Webpagina

Automatisch gegenereerde beschrijving

Misschien dat we daar nog wel kunnen bekijken of die Dockerfile niet beter kan.

Docker compose:

Eerst version declareren

Dan onder services alles oplijsten wat moet runnen. In ons geval 1 mongoDB container (mongo-products) en 3 mysql containers (mysql-order, mysql-inventory, mysql-user). Die hebben allen een healthcheck en pas nadat deze voltooid is, zouden de eigenlijke services verbinding mogen/kunnen maken.

Die services zijn dan simpelweg product-service, inventory-service, user-service, order-service.

Van elke service wordt eerst de image gepulld die op dockerhub staat. Nadien bij depends on wordt duidelijk gemaakt welke service er ook moet lopen vooraleer de service mag opstarten. Bij degene die sql gebruiken staat er nog een hele environment bij met username, paswoord en met welke db je connecteert.

MYSQL\_DB\_PORT moet bij elke service op 3306 staan, omdat het in containers samen runt. Dit is dus anders als je alles apart runt blijkbaar (staat op Canvas bij uitleg Michiel). Bij sommige staat nog een baseurl voor andere services waarmee ze moeten communiceren.

Als laatste enkele volumes voor iedere db.

In principe moet hier nix meer aan gebeuren.

De services:

Allemaal ongeveer hetzelfde opgebouwd.

User-service is volledig nieuw, gebruikt vier DTOs (of gebruikte). Heb enkele methodes terug in comment gezet, omdat ik niet zeker wist of we ze nog nodig zouden hebben of niet.

De userrepository wordt in de userservice geïnjecteerd en de userservice wordt in de usercontroller geïnjecteerd.

Dit gebeurt via @RequiredArgsConstructer & private final UserService userservice. Aparte code om dit te injecteren is dus niet meer nodig (denk aan this.userservice = userservice en zo verder; zie cursus Java Advanced of ASP.NET :P )

Veel meer valt er voorlopig niet te vertellen denk ik, dus best vragen dan!