

Johannes Fog

Gruppemedlemmer:

Navn: Abdullah Marwan El-Haj Moussa

Email: cph-ae165@cphbusiness.dk

Github: AbdullahMarwan

Hold: I-dat-da 0222a - Sem2

Navn: Mohammad Marwan El-Haj Moussa

Email: cph-me283@cphbusiness.dk

Github Mohammadx9000

Hold: I-dat-da 0222a - Sem2

Torsdag d.??/12 - 2022

Indholdsfortegnelse

Links	3
Indledning	4
Baggrund	4
Teknologivalg	4
Valg af arkitektur	5
Arbejdsprocess	5
SWOT analyse	5
Value proposition canvas	6
Krav	7
User stories (funktionelle krav)	7
Ikke funktionelle krav	8
Kanban board	8
Domain model	10
EER Diagram	10
CI/CD Pipeline	11
Docker	11
Enviroment variables	12
Encryption	15
Code improvements	16
Hibernation	16
getMaterials()	16
createUser()	17
Arbejdsprocess faktiskt	18
Arbejdsprocess reflekteret	18

Links

1. Her er linket til vores Github repository:
<https://github.com/AbdullahMarwan/CPH-Fog-Project-Semester2>
2. Her er linket til vores hjemmeside: (det kan være at der forekommer fejl)
<http://164.90.212.208:4100/fog-1.0-SNAPSHOT/>
 - a. **Login:**
 - i. Username: "admin"
 - ii. Password: "1234"
3. Demo video: <https://youtu.be/UJvXU0BpFCA>
4. Pitch video: <https://youtu.be/RpES0sl5qRc>

Indledning

Dette projekt handler om at lave en hjemmeside til firmaet Fog, hvor kunden skal kunne ordre/bestille en carport, ved kun at indsætte 3 værdier, længde, bredde og om kunden vil have et skur. Til at opnå dette mål, har gruppen brugt diverse programmer og værktøjer, såsom IntelliJ, som IDE, Digitalocean som server og MySQL som database.

Baggrund

Vi har fået udleveret en opgave af firmaet Johannes Fog der arbejder med at konstruere og udlevere carporte til deres kunder. For at gøre det letter for deres medarbejdere at skitsere sig frem til kundens selvmålet carport. Kunden skal kunne bestille en carport ved at vælge længde, bredde og om at kunden vil have et skur med. Ud fra disse informationer skal hjemmesiden regne ud hvilke type og antal materialer der skal bruges til at konstruere den valgte carport. Der skal også udleveres en detaljeret materialeliste, samt en blueprint der viser hvordan carporten skal se ud.

Teknologivalg

Programmer og udviklingsværktøjer:

- IntelliJ IDEA Ultimate 2021.3.1
- GIT og Github
- Java SDK 18
- Tomcat Webcontainer 9.0.67
- MySql Server 8.0.30
- MySql Workbench 8.0.18
- Digital Ocean
- Docker
- Buddyworks Pipeline

Programmeringssprog, biblioteker med mere:

- Java / JSP / JSTL
- HTML 5
- CSS 3.0
- Twitter Bootstrap 5.0

Planlægning og rapportskrivning:

- Google Docs
- Draw.io til diagrammer
- Github – til versionering og deling af kode

Valg af arkitektur

Den valgte arkitektur i dette projekt er MVC som står for Model-View-Controller. Denne arkitektur deler koden i tre mindre dele og gøre det lettere for at visuelt skældne hvilke dele af koden hører til hvad.

Arbejdsprocess

Vi har arbejdet på projektet i følgende rækkefølge:

- Swot analyse
- VPC
- Domain Model
- User Stories
- Tasks I KanBan Modellen
- EER Diagram

SWOT analyse

Gruppen lavede en SWOT analyse af kundens firma, Fog, for at kunne bruge styrkerne i kunden til vores fordel, samt arbejde på svaghederne og truslerne så kunden kan få et rimelig resultat/produkt.

STRENGTHS	WEAKNESSES
<ul style="list-style-type: none">• Kendt brand• God materiale kvalitet	<ul style="list-style-type: none">• For mange parameter/tal for regnestykket• Kunder ved ofte ikke hvor mange materialer der skal købes
OPPORTUNITIES	THREATS
<ul style="list-style-type: none">• Ved at gøre processen nemmere, kan fog kan blive en mere attraktiv valg for kunder fremfor andre firmaer	<ul style="list-style-type: none">• Kunder går til andre firmaer som Silvan

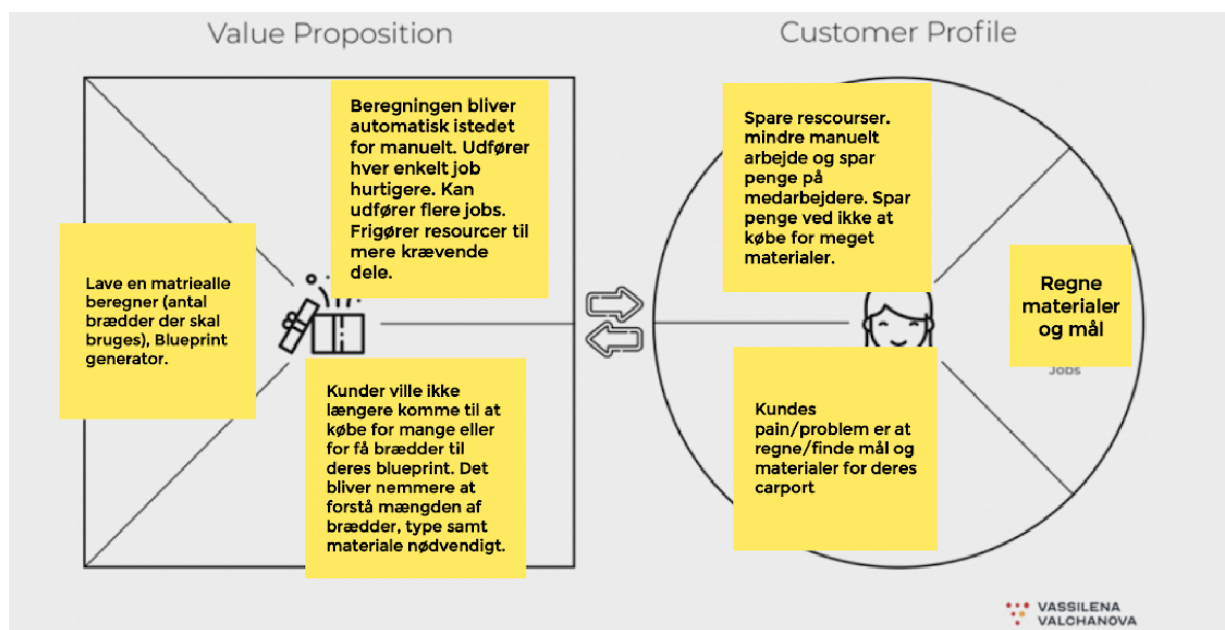
Value proposition canvas

Customer profile:

- **Customer Jobs:** Regne matrieller, mål,
- **Pain:** Kundes pain/problem er at regne/finde mål og materialer for deres carport
- **Gain:** Spare resourcer. mindre manuelt arbejde og spar penge på medarbejdere. Spar penge ved ikke at købe for meget materialer.

Value map:

- **Products and Services:** Lave en matrielle beregner (antal brædder der skal bruges), Blueprint generator.
- **Pain Relievers:** Kunder ville ikke længere komme til at købe for mange eller for få brædder til deres blueprint. Det bliver nemmere at forstå mængden af brædder, type samt materiale nødvendigt.
- **Gain Creators:** Beregningen bliver automatisk istedet for manuelt. Udfører hver enkelt job hurtigere. Kan udfører flere jobs. Frigører resourcer til mere krævende dele.



Krav

User stories (funktionelle krav)

- US-1: Som kunde kan jeg oprette en konto/profil for at kunne bestille en carport.
 - Givet at jeg indtaster min email og et kodeord i en formular, når jeg klikker på sign-up knappen, så bliver jeg registreret i databasen og kan senere logge ind for at bestille carporte.
 - Estimat: small
- US-2: Som kunde eller administrator kan jeg logge på systemet. Når jeg er logget på skal jeg kunne se de forskellige materialer der er tilgængelige i lageret.
 - Givet at jeg indtaster min email og et kodeord i en formular, når jeg klikker på login knappen, så bliver jeg logget ind og kan derefter bestille carporte.
 - Estimat: medium
- US-3: Som administrator kan jeg tilføje samt fjerne materialer i databasen eller ændre på deres værdier.
 - Givet at jeg er logget ind som administrator, når jeg er inde på databasen, så kan jeg tilføje/fjerne materialer fra databasen.
 - Estimat: medium
- US-4: Som kunde kan jeg se mine valgte overordnet carport dele i en indkøbskurv, så jeg kan se den samlede pris. (F.eks. tag)
 - Givet at jeg er logget ind som kunde, når jeg klikker på indkøbskurven, så kan jeg se min valgte carport dele.
 - Estimat: medium
- US-5: Som administrator kan jeg se alle ordrer i systemet, sådan at kan udføre forskellige handlinger med dem. (Fjerne, opdatere)
 - Givet at jeg er logget ind som administrator, når jeg klikker på odrer knappen (som kun vises til dem der er logget ind som administrator), så kan jeg se alle ordrerne i systemet og kan dermed fjerne eller opdatere dem.
 - Estimat: medium
- US-6: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.
 - Givet at jeg er logget ind som administrator, når jeg klikker på slet ordrer, så bliver den fjernet fra databasen.
 - Estimat: small
- US-7: Som administrator så kan jeg klikke på en ordre og se dens konkrete produkter. (F.eks. antal skruer, brædder osv)

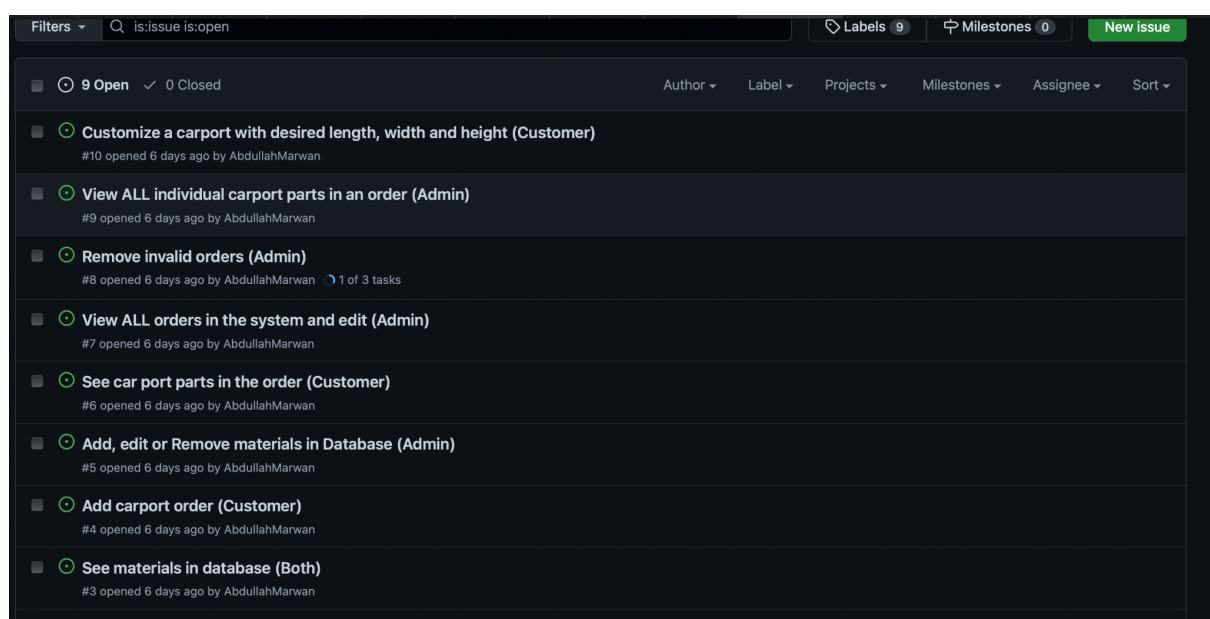
- Givet at jeg er logget ind som administrator, når jeg klikker på en ordre, vises der den fulde liste af materialer
- Estimat: medium
- US-8: Som kunde kan jeg bestille og betale carport med en valgfri længde og bredde, sådan at de kører forbi senere og bygger den.
 - Givet at jeg er logget ind som kunde, når jeg betaler for carporten efter at have indsat længde og bredde, så bliver den bestilt og der kommer medarbejder som så bygger den senere.
 - Estimat: large

Ikke funktionelle krav

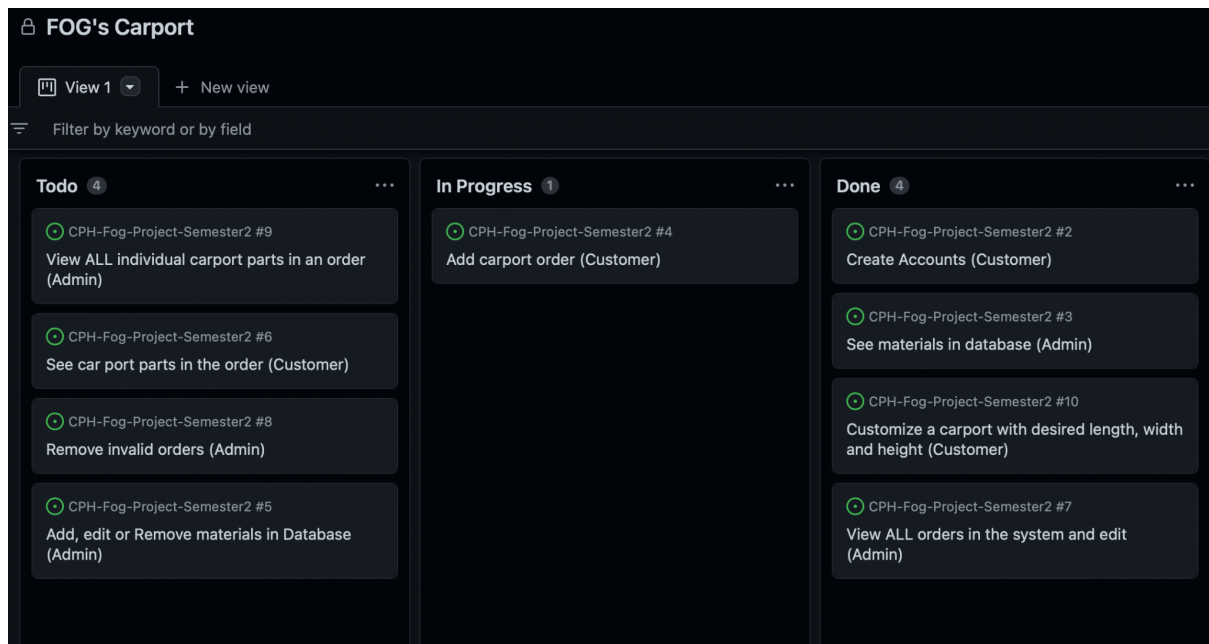
1. Løsning skal baseres på en MySQL 8 database.
2. Designet skal afspejle en flerlagsarkitektur, der afvikles på en Java server (Tomcat 9). I gør klogt i at tage udgangspunkt i den udleverede startcode.
3. Webapplikationen bør bygges af almindelige Java klasser, servlets, og JSP-sider.
4. Websiderne skal fungere i enten Google Chrome eller Firefox.
5. Websitet skal deployes i skyen (Digital Ocean)
6. Kildekoden skal være tilgængelig i et GitHub repository.

Kanban board

Vi har taget vores user stories og lavede dem om til issues som vi så har listed op på vores github repository:

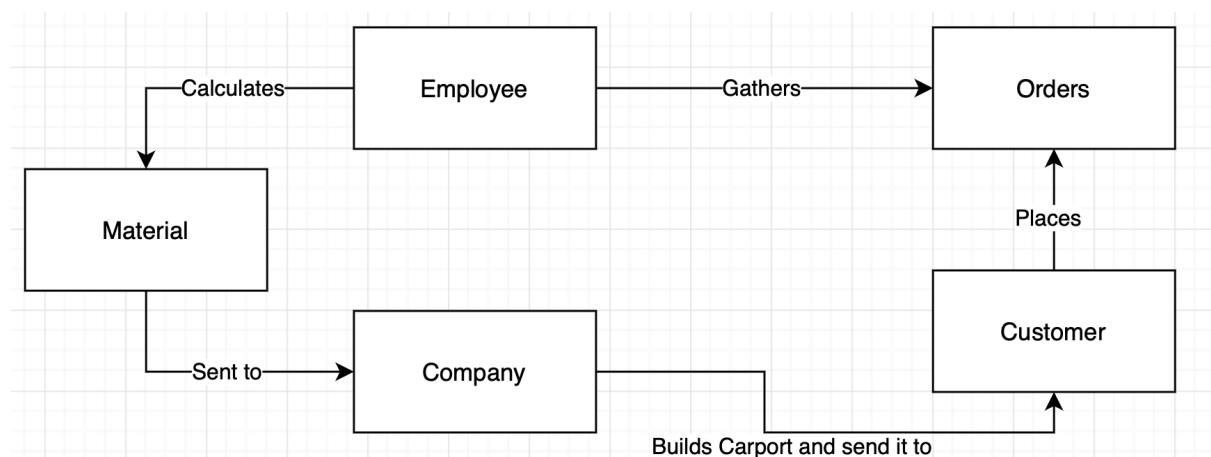


Derefter indsættede vi disse issues i et kanban board som giver os et bedre overblik af vores tasks. Følgende er det nuværende process på de forskellige tasks:



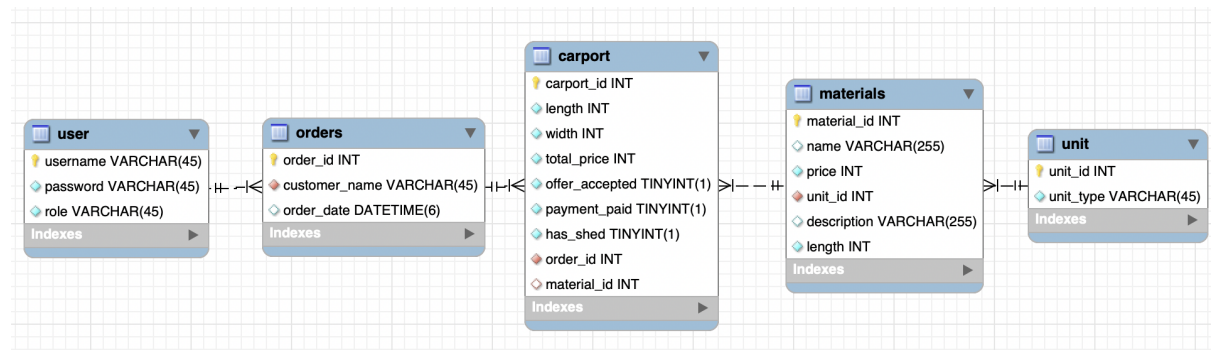
Domain model

Der er fem model elementer i dette domain model, som er Employee, Material, Company, Customer or Orders. En Customer placere en Ordre på websiden. En af medarbejderne samler alle ordrene og udregner en stykliste, samt en tegning af carporten. Disse bliver sendt videre til firmaet, der så forbereder materialerne og sender dem sammen med nogle medarbejdere til kundens hus, hvor de så bygger carporten.



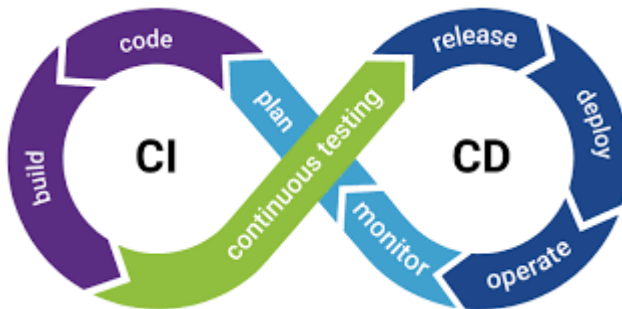
EER Diagram

MySQL databasen er bygget på følgende måde. User tabellen indeholder informationer om brugeren som navn og kodeord, navnet bliver forbundet med orders tabellen, som indeholder ordre id, dato og kundens navn, ordre id er forbundet til carport. I carport tabellen er der mål, pris og materiale id som er forbundet til materiale tabellen. Der ligger der navn, pris, mål og unit id der forbinder os til den sidste tabel, som er unit. Unit bruges til at identificere hvilke træ høre til hvilke part/dele, såsom hvis træet hører til taget på carporten eller siderne.

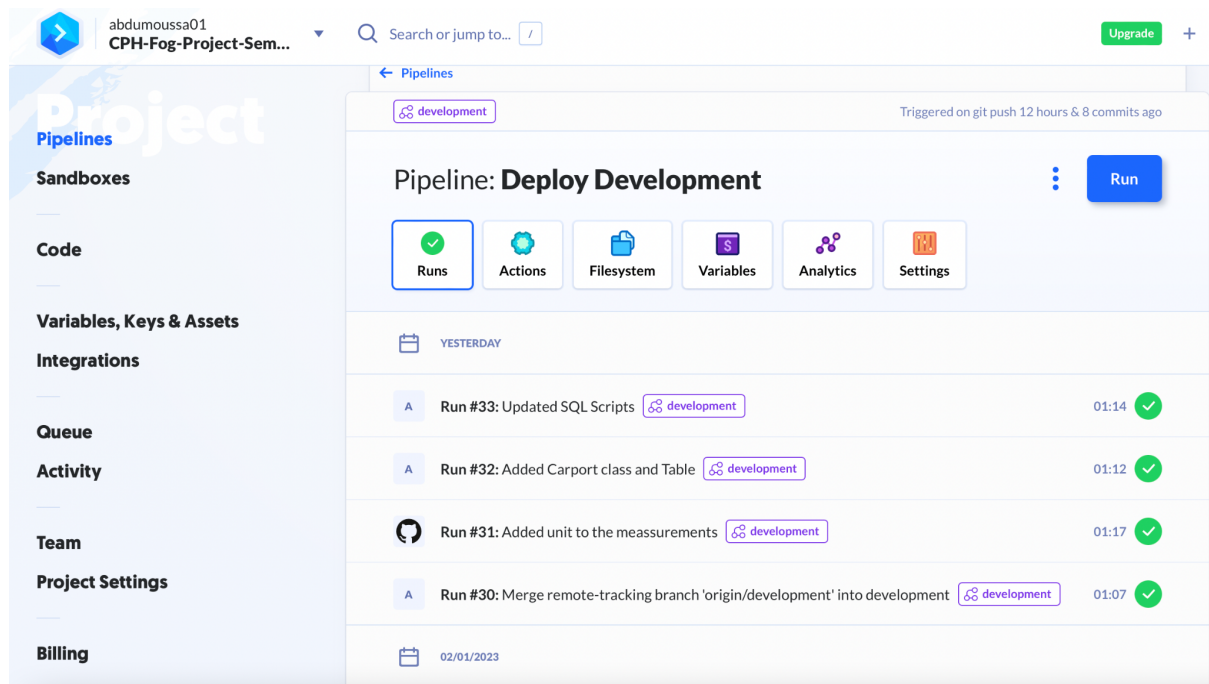


CI/CD Pipeline

En stor del af dette projekt er gjort nemmere ved brug af pipelines. Istedet for at manuelt uploade samt opdatere vores droplet hvert eneste gang der komittes en lille eller en stor ændring, bliver det hele full automateseret ved brug af pipelines. Dette kaldes for CI-CD der står for Continuous Integration and Continuous Delivery:



Ved brug af Buddyworks har vi sat dropletten til at opdatere samt deploye den valgte git branch hver gang der komittes noget til den valgte branch:



Docker

I dette projekt gør vi brug af Docker. ()

Denne proces kan forenkles ved hjælp af følgende diagram:



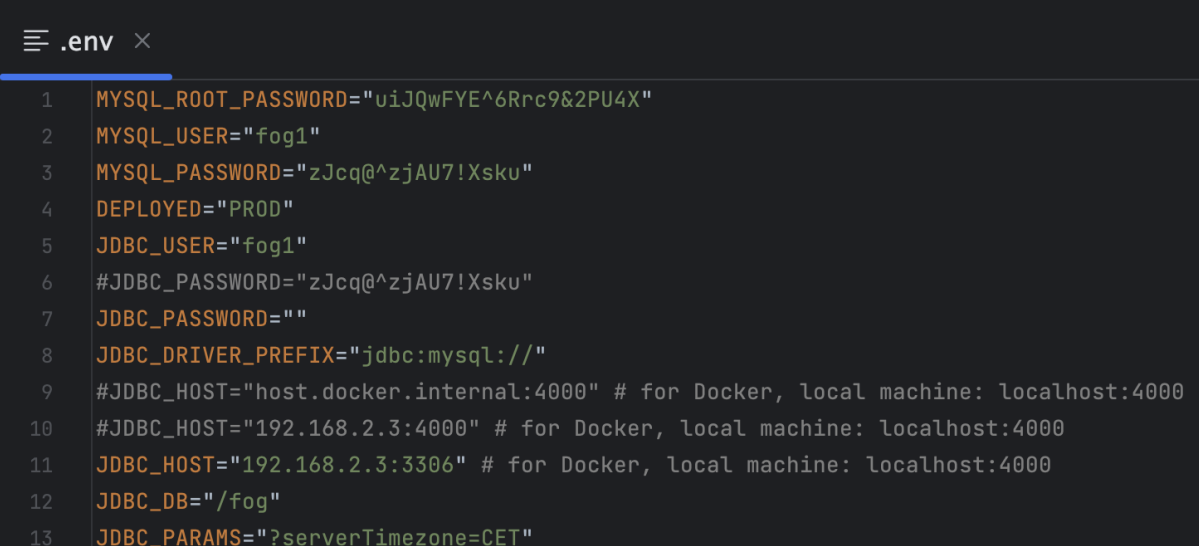
Dockerfilen er en tekstfil som indeholder alle de kommandoer programmet skal køre.

```
Dockerfile x
1 # Maven stage
2 FROM maven:3.6.0-jdk-11-slim AS mavenbuild
3 WORKDIR /home/app
4 COPY pom.xml /home/app/pom.xml
5 COPY src /home/app/src
6
7 #RUN mvn -v -X -f pom.xml clean package - DEBUG
8 RUN mvn -f pom.xml clean package
9
10 # Tomcat stage
11 FROM tomcat:9.0.70-jdk11 as tomcatbuild
12
13 COPY --from=mavenbuild /home/app/target/fog-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/
14
15 EXPOSE 8080
16
17 CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]
```

Docker image virker som en template der fortæller step by step hvordan projektet skal bygges når den køres.

Enviroment variables

For at gøre vores projekt mere sikkert så ingen uvedkommende kan få adgang til de sensitive oplysninger i projektet som bruger og kodeord til databasen, har vi gemt disse oplysninger i en .env fil som kun udviklerne har adgang til.

```
1 MYSQL_ROOT_PASSWORD="uiJQwFYE^6Rrc9&2PU4X"
2 MYSQL_USER="fog1"
3 MYSQL_PASSWORD="zJcq@^zjAU7!Xsku"
4 DEPLOYED="PROD"
5 JDBC_USER="fog1"
6 #JDBC_PASSWORD="zJcq@^zjAU7!Xsku"
7 JDBC_PASSWORD=""
8 JDBC_DRIVER_PREFIX="jdbc:mysql://"
9 #JDBC_HOST="host.docker.internal:4000" # for Docker, local machine: localhost:4000
10 #JDBC_HOST="192.168.2.3:4000" # for Docker, local machine: localhost:4000
11 JDBC_HOST="192.168.2.3:3306" # for Docker, local machine: localhost:4000
12 JDBC_DB="/fog"
13 JDBC_PARAMS="?serverTimezone=CET"
```

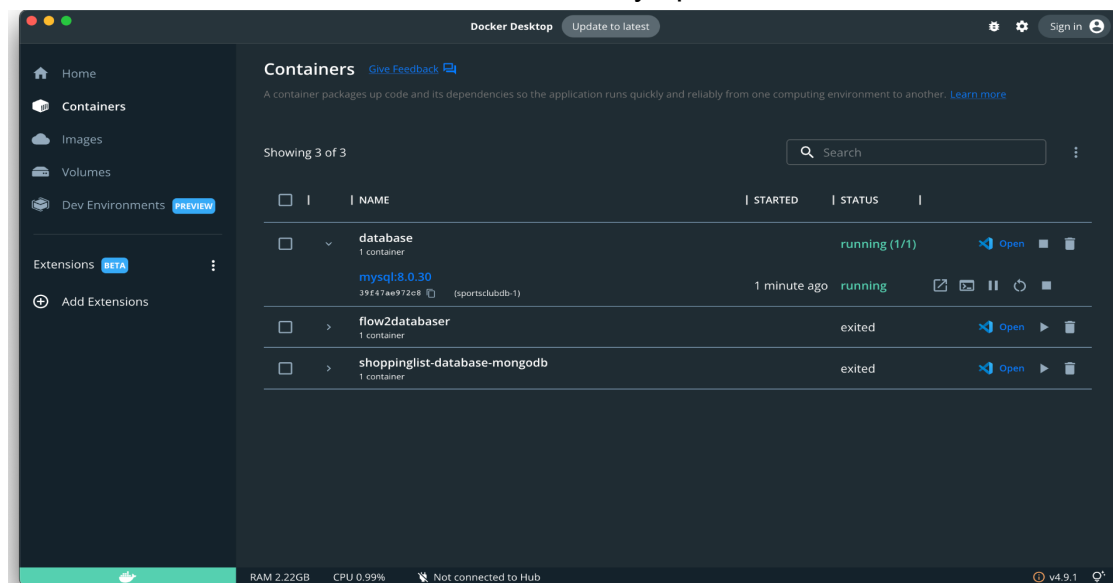
Dockercompose

Dockercompose filen indeholder projektet's image, hvor alle connection værdierne til vores database samt hjemmeside bliver brugt. For at gøre disse sensitive informationer sikre, har vi gemt værdierne i en environment fil som kun udviklerne har adgang til.

```
docker-compose.yml x
1  version: '3.8'
2  >> services:
3  > fogdb:
4      image: mysql:8.0.30
5      #restart: always
6      restart: on-failure
7      environment:
8          - MYSQL_DATABASE=fog
9          - MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD
10         - MYSQL_USER=$MYSQL_USER
11         - MYSQL_PASSWORD=$MYSQL_PASSWORD
12     ports:
13         - '4000:3306'
14     networks:
15         fogprojektnetwork:
16             ipv4_address: 192.168.2.3
17     volumes:
18         - fogdb:/var/lib/mysql
19         - fogdb-log:/var/log/mysql
20         - ./database/scripts/structure_data:/docker-entrypoint-initdb.d
21 > fogweb:
22     depends_on:
23         - fogdb
24     build:
25         context: .
26         dockerfile: Dockerfile
27     image: tomcat
28     environment:
29         - DEPLOYED=$DEPLOYED
30         - JDBC_USER=$JDBC_USER
31         - JDBC_PASSWORD=$JDBC_PASSWORD
32         - JDBC_DRIVER_PREFIX=$JDBC_DRIVER_PREFIX
33         - JDBC_HOST=$JDBC_HOST
34         - JDBC_DB=$JDBC_DB
```

```
34     - JDBC_DB=$JDBC_DB
35     - JDBC_PARAMS=$JDBC_PARAMS
36   ports:
37     - '4100:8080'
38   networks:
39     fogprojektnetwork:
40       ipv4_address: 192.168.2.2
41   extra_hosts:
42     - 'host.docker.internal:192.168.2.1' #- 'host.docker.internal:host-gateway'
43   volumes:
44     fogdb:
45       driver: local
46     fogdb-log:
47       driver: local
48   networks:
49     fogprojektnetwork:
50       driver: bridge
51     ipam:
52       driver: default
53       config:
54         - subnet: "192.168.2.0/24"
55           gateway: "192.168.2.1"
```

Her ses vores Docker Container der kører mysql databasen:



Encryption

På nuværende tidspunkt anvender vi ingen kryptering på selve websitet, men det er noget meget vigtigt vi har tænkt os at tilføje senere hen.

Code improvements

I dette projekt har vi forbedret den allerede givet startkode ved brug af flere metoder. Her er givet et par eksempler på før og efter sammenligning.

Hibernation

getMaterials()

Den gamle getMaterials metoden der gør brug af connection pool og preparedstatements fyldte en meget stor del. Connection Pool er blevet erstatet med JDBC connection.

Hvis man skulle opdatere tabellen eller ændre på variabel typen på et eller flere af variablerne ville man blive nødt til næsten at gøre hele processen om igen manuelt, dette er meget tidskrevende.

```
static List<Material>
OLDgetMaterials(dat.backend.model.persistence.ConnectionPool connectionPool) {
    List<Material> materialList = new ArrayList<>();

    String sql = "select * from materials";

    try (Connection connection = connectionPool.getConnection()) {
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("material_id");
                String name = rs.getString("name");
                int price = rs.getInt("price");
                int unit_id = rs.getInt("unit_id");
                String description = rs.getString("description");
                int length = rs.getInt("length");

                Material newMaterial = new Material(id, name, price, unit_id,
description, length);
                materialList.add(newMaterial);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return materialList;
}
```


Dette problem er blevet løst og processen forenklet med den nye getMaterials metode som gør brug af Hibernate som gør at koden er mere enkel og fylder mindre. Med Hibernate så opdatere og loader man hele objekter.

```
static List<Material> newGetMaterials() {
    Logger.getLogger("web").log(Level.INFO, "");

    PersistenceManager persistenceManager = new PersistenceManager("fog-unit");

    TypedQuery<Material> query =
    persistenceManager.getEntityManager().createNamedQuery("getAllMaterials",
    Material.class);

    return query.getResultList();
}
```

createUser()

Det samme mønster i getMaterials() afsnittet hvor vi læser objekter fra databasen kan ses i createUser() metoderne hvor vi gemmer objekter i databasen. Med den gamle metode skulle der indtastes ethvert værdi som gjorde at det ikke nemt kan ændres og at det fylder meget mere.

```
static User OLDcreateUser(String username, String password, String role,
    ConnectionPool connectionPool) throws DatabaseException {
    Logger.getLogger("web").log(Level.INFO, "");
    User user;
    String sql = "insert into user (username, password, role) values (?, ?, ?)";
    try (Connection connection = connectionPool.getConnection()) {
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, username);
            ps.setString(2, password);
            ps.setString(3, role);
            int rowsAffected = ps.executeUpdate();
            if (rowsAffected == 1) {
                user = new User(username, password, role);
            } else {
                throw new DatabaseException("The user with username = " +
                username + " could not be inserted into the database");
            }
        }
    } catch (SQLException ex) {
        throw new DatabaseException(ex, "Could not insert username into
        database");
    }
    return user;
}
```

Med den nye metode kan vi med brug af java hibernate, nemt gemme hele objekter i databasen kun ved brug af meget få linjer som slet ikke skal opdateres hvis der bliver lavet ændringer på variableerne:

```
static User newUser(User user) {
    Logger.getLogger("web").log(Level.INFO, "");

    PersistenceManager persistenceManager = new
    PersistenceManager("fog-unit");

    persistenceManager.entityTransaction().begin();

    persistenceManager.getEntityManager().persist((user));
    persistenceManager.entityTransaction().commit();

    return null;
}
```

Arbejdsprocess faktuel

- Der blev arbejdet og færdiggjort på fire user stories, som er US-1, US-2, US- og US-8
- Kanban Mesteren (KM) i løbet af dette projekt var Abdullah. Som KB gav han tasks til de andre medlemmer af gruppen (Mohammad), som f.eks. at arbejde på US-8, imens han arbejdede på US-1.

Arbejdsprocess reflekteret

Gruppen har underestimeret sværhedsgraden/hvor meget tidskrævende opgaverne egentlig var, og dermed endte gruppen med kun at have gennemført halvdelen af de angivet user stories