

User Story 2: Opsæt Docker build og runtime

Formål

At kunne bygge og køre AdventureXP projektet i Docker containere, så det kan deployes på alle maskiner uafhængigt af lokalt miljø.

Task 2.1: Udarbejd multi-stage Dockerfile

Implementering:

Vi har oprettet en multi-stage Dockerfile der indeholder disse 2 punkter

- **Build stage**, som bruger `openjdk:17-jdk-slim` til at bygge projektet med Maven
- **Production stage**, som bruger `eclipse-temurin:17-jre` til at køre applikationen (mindre image)


Sikkerhedsfunktioner:

- Non-root user (`appuser`) kører applikationen
- Health check endpoint på `/api/activities`
- Minimal runtime image (JRE i stedet for JDK)

Dockerfile strukturen:

Det komplette Dockerfile kan ses i vores projekt repository,

Dette er fra vores kode som vi bruger til at bygge projektet med maven

```
# Multi-stage build for optimal image size
FROM openjdk:17-jdk-slim as  build
```

Den her bruges til produktionen til at køre vores applikation

```
# Production stage
FROM eclipse-temurin:17-jre
```

Task 2.2: Konfigurerer docker-compose.yaml

Services

Vores docker-compose.yaml definerer 3 services:

1. MySQL Database (mysql)

- Image: mysql:8.0
- Port: 3306 (intern)
- Database: adventurepark
- Credentials: appuser / apppassword
- Health check: Tjekker MySQL er klar før app starter
- Volume: mysql_data for data persistence

2. Spring Boot App (app)

- Bygges fra vores Dockerfile
- Port: 8080 (eksponeret)
- Profile: docker
- Depends on: MySQL (med health check condition)
- Environment variables til database forbindelse

3. Adminer (adminer)

- Image: adminer:latest
- Port: 8081 (eksponeret)
- Database management interface

Netværk og Volumes

- Netværk: adventure-park-network (bridge driver)
- Volume: mysql_data (persisterer database data)

Task 2.3: Test lokal opstart af systemet via Compose

1. Start systemet

Kør denne kode i terminalen

```
docker compose up --build
```

Det forventede output burde være

- ✓ Container adventure-park-db Started
- ✓ Container adventure-park-adminer Started
- ✓ Container adventure-park-app Started

2. Verificer container status

Kør denne kode i terminalen

```
docker compose ps
```

Resultat:

- Alle 3 containere skal være "Up"
- app og db skal være "(healthy)"

```
PS C:\Users\Heilstrup\IdeaProjects\adventure-fullstack> docker compose ps
NAME                IMAGE                COMMAND                SERVICE    CREATED        STATUS
adventure-park-adminer  adminer:latest      "entrypoint.sh docke..." adminer    2 hours ago    Up 2 hours
adventure-park-app      adventure-fullstack-app "java -jar app.jar"    app       2 hours ago    Up 2 hours (healthy)
adventure-park-db       mysql:8.0            "docker-entrypoint.s..." mysql     2 hours ago    Up 2 hours (healthy)
```

3. Test API Endpoints

Hent aktiviteter:

Hent aktiviteter

Kør disse kommandoer i terminalen

```
curl http://localhost:8080/api/activities
```

Initialiser test data:

```
curl -X POST http://localhost:8080/api/activities/init-data
```

Verificer data:

```
curl http://localhost:8080/api/activities
```

Resultatet burde derefter returnere 4 aktiviteter (Go-kart, Minigolf, Paintball og Sumo Wrestling)

4. Test Frontend

URL: <http://localhost:8080>

Funktionalitet testet:

- Visning af aktiviteter
- Filtrering efter alder og antal deltagere
- Booking system
- Reservation oversigt

Se Screenshot 2 for frontend interface.

Go-kart

Spændende go-kart bane

Min. alder: 12 år

Max deltagere: 8

Varighed: 15 min

Pris: 150 DKK

Udstyr: Krævet

Minigolf

18 hullers minigolf bane

Min. alder: 5 år

Max deltagere: 10

Varighed: 60 min

Pris: 75 DKK

Udstyr: Ikke nødvendigt

Paintball

Taktisk paintball kamp

Min. alder: 16 år

Max deltagere: 12

Varighed: 90 min

Pris: 200 DKK

Udstyr: Krævet

Sumo Wrestling

Sjov sumo brydning

Min. alder: 8 år

Max deltagere: 4

Varighed: 20 min

Pris: 100 DKK

Udstyr: Krævet

Login credentials:

- Server: mysql
- Username: appuser
- Password: apppassword
- Database: adventurepark

Verificeret:

- activities tabel indeholder 4 rækker
- reservations tabel eksisterer
- Data persisterer korrekt

Se Screenshot 3 for database indhold.

The screenshot shows the Adminer web interface (version 5.4.0 5.4.1) connected to a MySQL database named 'adventurepark'. The interface is in Danish. The 'SQL-kommando' (SQL command) tab is active, displaying the query: `SELECT * FROM activities`. The result shows 4 rows of data from the 'activities' table.

id	description	duration_minutes	equipment_required	max_participants	min_age	name	price
1	Spændende go-kart bane	15	1	8	12	Go-kart	150
2	18 hullers minigolf bane	60	0	10	5	Minigolf	75
3	Taktisk paintball kamp	90	1	12	16	Paintball	200
4	Sjov sumo brydning	20	1	4	8	Sumo Wrestling	100

Below the table, it indicates '4 rækker (0.000 s) Rediger, Explain, Eksport' (4 rows (0.000 s) Edit, Explain, Export). At the bottom, there are controls for 'Kør' (Run), 'Limit rows', 'Stop ved fejl' (Stop on error), 'Vis kun fejl' (Show only errors), and a 'Historik' (History) button.

6. Test Data Persistence

Stop containere

`docker compose down`

Start igen

`docker compose up`

Verificer data stadig findes

`curl http://localhost:8080/api/activities`

Kommandooversigt over docker

Starter systemet

`docker compose up --build` # Build og start

`docker compose up -d` # Start i baggrunden

`docker compose logs -f app` # Se logs

Stopper systemet

`docker compose down` # Stop containere

`docker compose down -v` # Stop og slet volumes (data)

Debugger systemet

`docker compose down` # Stop containere

`docker compose down -v` # Stop og slet volumes (data)

Tester systemet

`curl http://localhost:8080/api/activities`

`curl -X POST http://localhost:8080/api/activities/init-data`

