Software Technology Project

Final Presentation

…

PollHub

Gruppe 9

FeedApp that became PollHub

Overview of key technologies

- Brief Frontend (with demo)
- IoT Device Simulation (demo)
- Swagger Exposing the API
- Overview of the backend in Spring
- Docker for dev and deployment

Front-End

React:

Reason for choosing; Wanting to learn React

React Router: Used to enable URL navigation between components

React Redux:
Central store of states, used for easy access to states between components

Demo

IoT Device Simulation

Since we needed a demo IoT device,

Flutter was chosen due to its properties:

Cross platform development for
iOS, Android, macOS, Web, Windows, Linux with one codebase

Flutter consists of Widgets (HTMLish) and Dart language

Demo

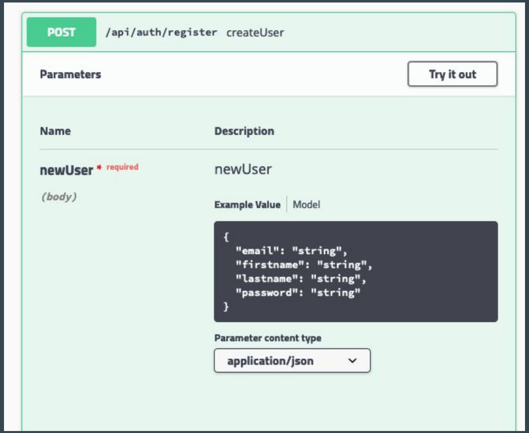
Swagger Exposing the API

Swagger did the following for us:

Overview of REST controller actions

Testing and verifying

Bug tracing



POST /api/auth/register createUser

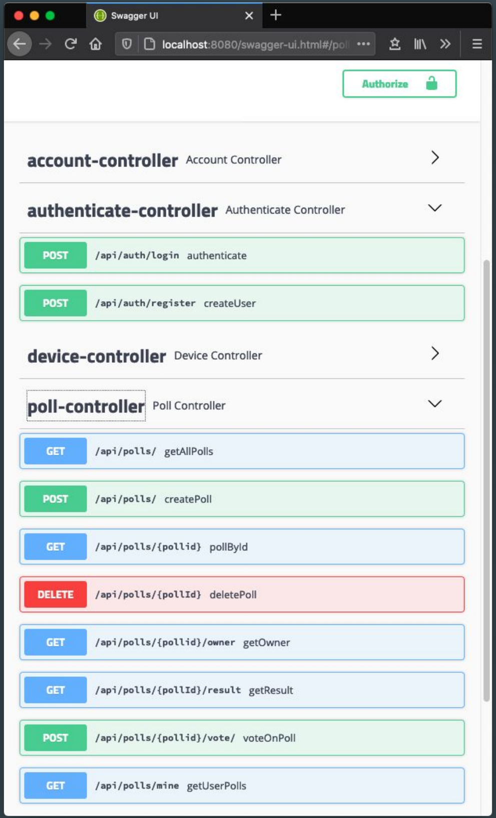
Parameters — Try it out

Name	Description
newUser * required (body)	newUser

Example Value | Model

```
{
  "email": "string",
  "firstname": "string",
  "lastname": "string",
  "password": "string"
}
```

Parameter content type: application/json



Swagger UI — localhost:8080/swagger-ui.html#/poll

Authorize

account-controller Account Controller

authenticate-controller Authenticate Controller

- POST /api/auth/login authenticate
- POST /api/auth/register createUser

device-controller Device Controller

poll-controller Poll Controller

- GET /api/polls/ getAllPolls
- POST /api/polls/ createPoll
- GET /api/polls/{pollid} pollById
- DELETE /api/polls/{pollId} deletePoll
- GET /api/polls/{pollid}/owner getOwner
- GET /api/polls/{pollId}/result getResult
- POST /api/polls/{pollid}/vote/ voteOnPoll
- GET /api/polls/mine getUserPolls

Overview of the backend in Spring

Spring framework is the core of our API

Made it easy to replace the database.
(e.g Derby => MySQL or any other)

Also made it easy to incorporate new services in the API e.g RabbitMQ.

Securing the API with Spring Security.

Overview of the backend in Spring

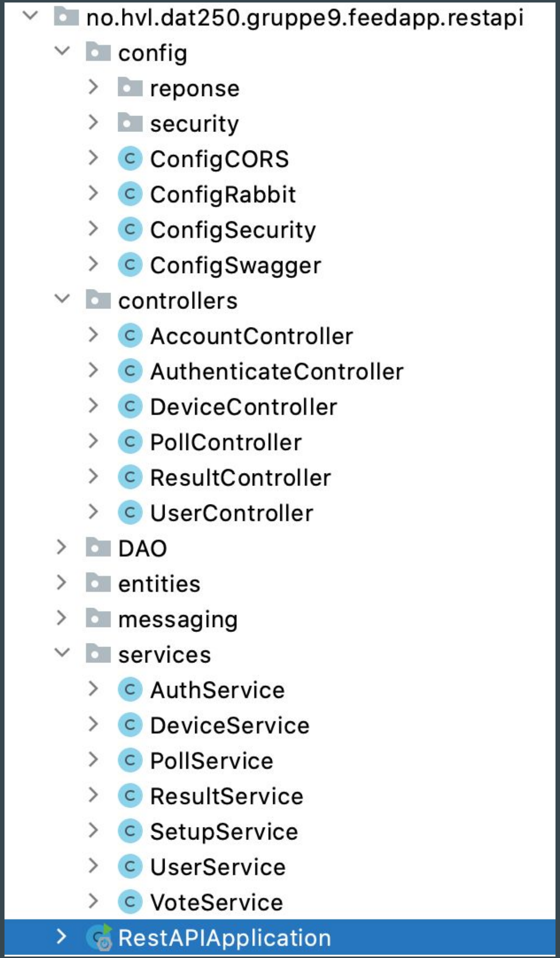
Controllers are thin,

only what need to be there is there.

Logic is in “Services”,

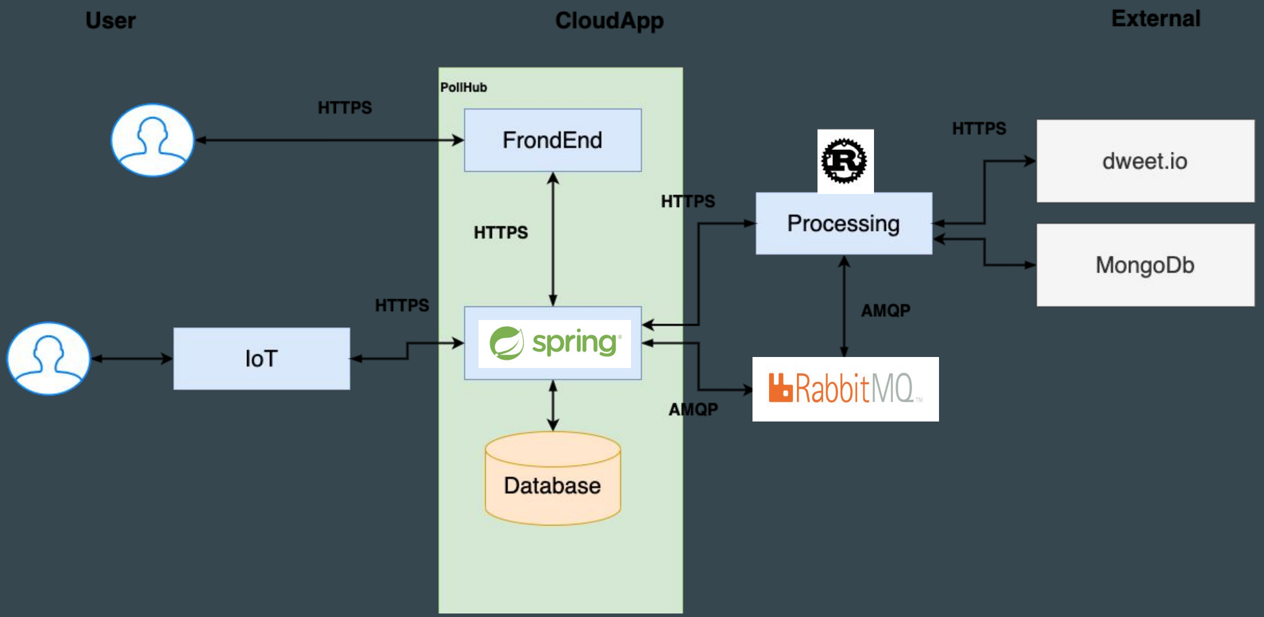
main logic of the operation.

DAO handles the storage/retrieving.



```
no.hvl.dat250.gruppe9.feedapp.restapi
  config
    reponse
    security
    ConfigCORS
    ConfigRabbit
    ConfigSecurity
    ConfigSwagger
  controllers
    AccountController
    AuthenticateController
    DeviceController
    PollController
    ResultController
    UserController
  DAO
  entities
  messaging
  services
    AuthService
    DeviceService
    PollService
    ResultService
    SetupService
    UserService
    VoteService
  RestAPIApplication
```

Overview of the backend in Spring

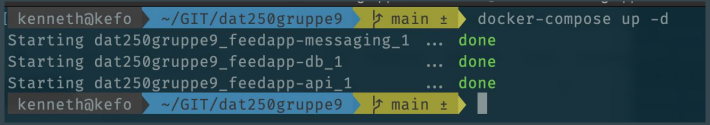


Docker for dev and deployment

Challenges

To run the API:
- The team had to install (MySQL on their machines)
- Later it became clear, they needed RabbitMQ also
- Also Microservice is in Rust, they need Rust Tools
- also need Maven/Java and so forth...

List is endless.

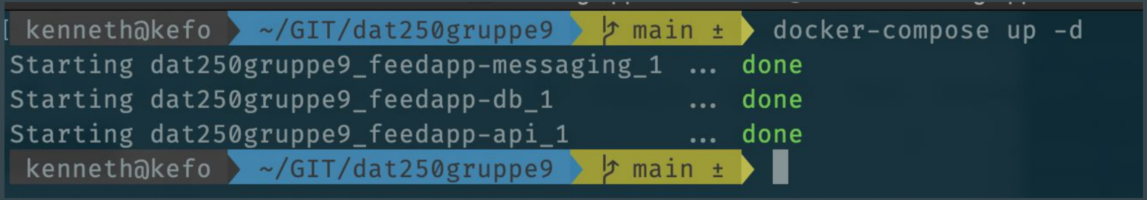


```
 kenneth@kefo  ~/GIT/dat250gruppe9  main ±  docker-compose up -d
Starting dat250gruppe9_feedapp-messaging_1 ... done
Starting dat250gruppe9_feedapp-db_1        ... done
Starting dat250gruppe9_feedapp-api_1       ... done
 kenneth@kefo  ~/GIT/dat250gruppe9  main ± 
```

Docker for dev and deployment

We dockerized the API and pulled the mysql:latest.

No, all they need to do is write the following command:

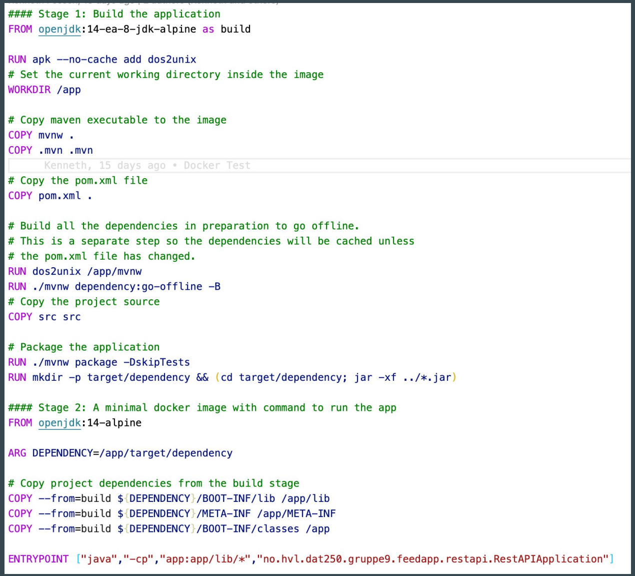


```
kenneth@kefo  ~/GIT/dat250gruppe9  main ±  docker-compose up -d
Starting dat250gruppe9_feedapp-messaging_1  ... done
Starting dat250gruppe9_feedapp-db_1         ... done
Starting dat250gruppe9_feedapp-api_1        ... done
kenneth@kefo  ~/GIT/dat250gruppe9  main ± 
```

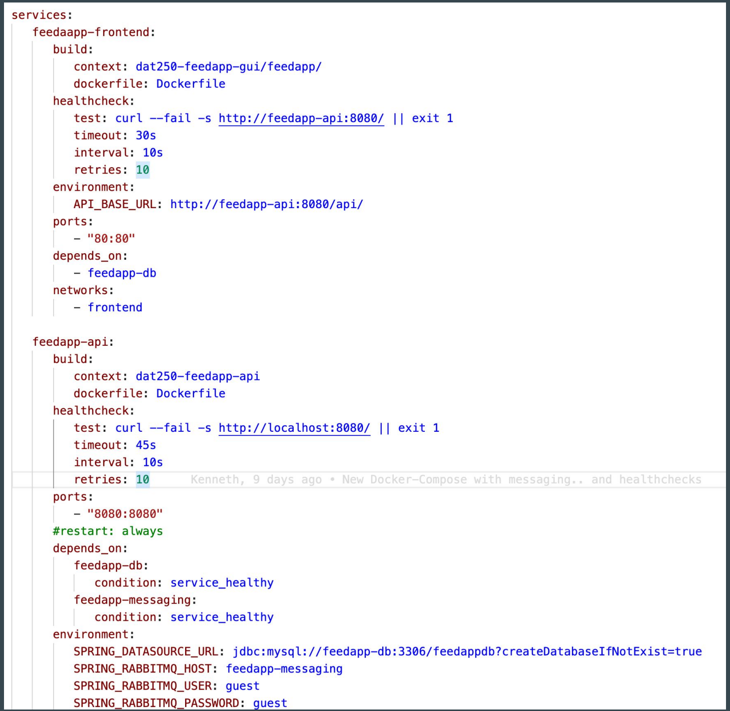
And the FrontEnd developer don’t have to install anything (except docker)

Docker for dev and deployment

Dockerfile / docker-compose.yml



```
#### Stage 1: Build the application
FROM openjdk:14-ea-8-jdk-alpine as build

RUN apk --no-cache add dos2unix
# Set the current working directory inside the image
WORKDIR /app

# Copy maven executable to the image
COPY mvnw .
COPY .mvn .mvn
      Kenneth, 15 days ago • Docker Test
# Copy the pom.xml file
COPY pom.xml .

# Build all the dependencies in preparation to go offline.
# This is a separate step so the dependencies will be cached unless
# the pom.xml file has changed.
RUN dos2unix /app/mvnw
RUN ./mvnw dependency:go-offline -B
# Copy the project source
COPY src src

# Package the application
RUN ./mvnw package -DskipTests
RUN mkdir -p target/dependency && (cd target/dependency; jar -xf ../*.jar)

#### Stage 2: A minimal docker image with command to run the app
FROM openjdk:14-alpine

ARG DEPENDENCY=/app/target/dependency

# Copy project dependencies from the build stage
COPY --from=build ${DEPENDENCY}/BOOT-INF/lib /app/lib
COPY --from=build ${DEPENDENCY}/META-INF /app/META-INF
COPY --from=build ${DEPENDENCY}/BOOT-INF/classes /app

ENTRYPOINT ["java","-cp","app:app/lib/*","no.hvl.dat250.gruppe9.feedapp.restapi.RestAPIApplication"]
```



```
services:
  feedaapp-frontend:
    build:
      context: dat250-feedapp-gui/feedapp/
      dockerfile: Dockerfile
    healthcheck:
      test: curl --fail -s http://feedapp-api:8080/ || exit 1
      timeout: 30s
      interval: 10s
      retries: 10
    environment:
      API_BASE_URL: http://feedapp-api:8080/api/
    ports:
      - "80:80"
    depends_on:
      - feedapp-db
    networks:
      - frontend

  feedapp-api:
    build:
      context: dat250-feedapp-api
      dockerfile: Dockerfile
    healthcheck:
      test: curl --fail -s http://localhost:8080/ || exit 1
      timeout: 45s
      interval: 10s
      retries: 10      Kenneth, 9 days ago • New Docker-Compose with messaging.. and healthchecks
    ports:
      - "8080:8080"
    #restart: always
    depends_on:
      feedapp-db:
        condition: service_healthy
      feedapp-messaging:
        condition: service_healthy
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://feedapp-db:3306/feedappdb?createDatabaseIfNotExist=true
      SPRING_RABBITMQ_HOST: feedapp-messaging
      SPRING_RABBITMQ_USER: guest
      SPRING_RABBITMQ_PASSWORD: guest
```

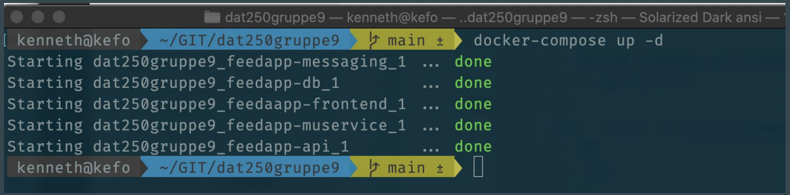
Docker for dev and deployment

Benefits:

- everyone in the group had same version of the software.
- no need to install.
- windows/mac/linux they all write the same command
- free to choose tech in the stack

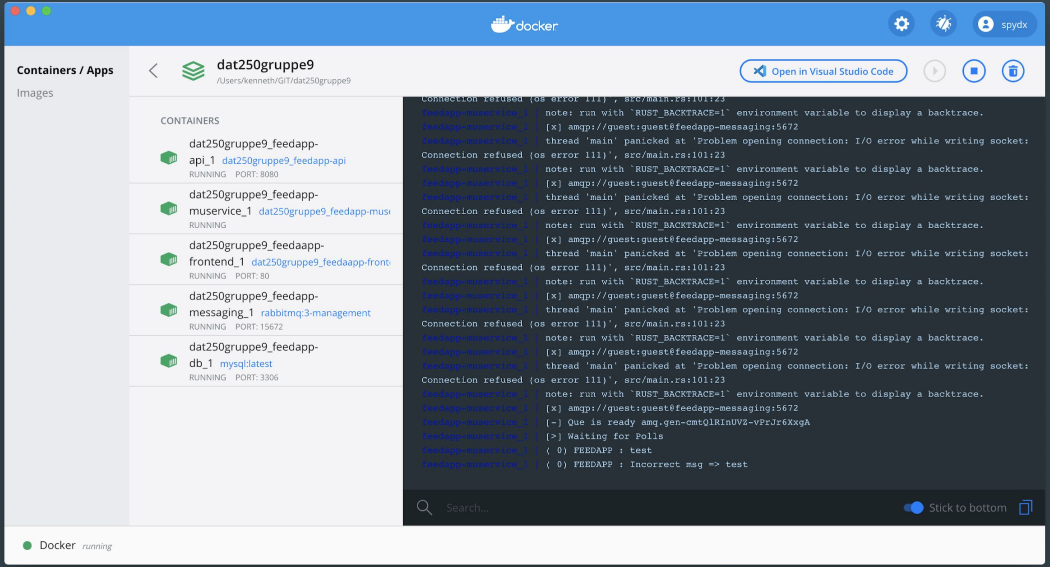
Docker for dev and deployment

Cmdline:



```
dat250gruppe9 — kenneth@kefo — ..dat250gruppe9 — -zsh — Solarized Dark ansi —
kenneth@kefo  ~/GIT/dat250gruppe9  main ±  docker-compose up -d
Starting dat250gruppe9_feedapp-messaging_1 ... done
Starting dat250gruppe9_feedapp-db_1        ... done
Starting dat250gruppe9_feedaapp-frontend_1 ... done
Starting dat250gruppe9_feedapp-muservice_1 ... done
Starting dat250gruppe9_feedapp-api_1       ... done
kenneth@kefo  ~/GIT/dat250gruppe9  main ± 
```

GUI



docker spydx

Containers / Apps
Images

dat250gruppe9
/Users/kenneth/GIT/dat250gruppe9

Open in Visual Studio Code

CONTAINERS

dat250gruppe9_feedapp-api_1 dat250gruppe9_feedapp-api
RUNNING PORT: 8080

dat250gruppe9_feedapp-muservice_1 dat250gruppe9_feedapp-muse
RUNNING

dat250gruppe9_feedaapp-frontend_1 dat250gruppe9_feedapp-front
RUNNING PORT: 80

dat250gruppe9_feedapp-messaging_1 rabbitmq:3-management
RUNNING PORT: 15672

dat250gruppe9_feedapp-db_1 mysql:latest
RUNNING PORT: 3306

```
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | thread 'main' panicked at 'Problem opening connection: I/O error while writing socket:
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | thread 'main' panicked at 'Problem opening connection: I/O error while writing socket:
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | thread 'main' panicked at 'Problem opening connection: I/O error while writing socket:
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | thread 'main' panicked at 'Problem opening connection: I/O error while writing socket:
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | thread 'main' panicked at 'Problem opening connection: I/O error while writing socket:
Connection refused (os error 111)', src/main.rs:101:23
feedapp-muservice_1 | note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace.
feedapp-muservice_1 | [x] amqp://guest:guest@feedapp-messaging:5672
feedapp-muservice_1 | [-] Que is ready amq.gen-cmtQlRInUVZ-vPrJr6XxgA
feedapp-muservice_1 | [>] Waiting for Polls
feedapp-muservice_1 | ( 0) FEEDAPP : test
feedapp-muservice_1 | ( 0) FEEDAPP : Incorrect msg => test
```

Search... Stick to bottom

Docker *running*

TechStack Overview

FrontEnd

React

React Redux

Backend

Swagger
Supported by SMARTBEAR

Java™

spring®

HIBERNATE

Db/Msg

RabbitMQ™

mongoDB®

MySQL™

Cloud

docker

Kubernetes Engine

Google Cloud Platform

IoT

App

Flutter

Hw

Project

- Andrè Frøseth Jønland
- Jan-Erik Erstad
- Kenneth Fossen
- Rune Almåsbakk

spydx/dat250gruppe9: Gruppe oppgave for DAT250

Links to relevant software

Flutter - Beautiful native apps in record time
Spring | Home
React – A JavaScript library for building user interfaces
Swagger: API Documentation & Design Tools for Teams
Docker: Empowering App Development for Developers
Rust Programming Language