Guide til GitPushers auktionsplatform

Anders, Frederik, Jacob & Jacob

Her er en step-by-step guide til at benytte vores auktionsplatform lokalt og de forskellige endpoints som skal benyttes for at udføre en auktion.

1. Opret Docker Netværk

Først logges der ind på din Docker bruger:

```
$ docker login
```

Derefter skal der oprettes et external bridge-netværk, som skal bruges til at binde de to Docker-compose filerne sammen.
Kør kommandoen:

```
$ docker network create sharednetwork
```

2. Docker Compose Filer

Start med at hente de følgende Docker compose filer i vores GitHub repository **DockerComposeFiles**:

- `docker-compose-services.yml`
- `docker-compose-aux.yml`

Kør derefter `docker-compose-aux.yml` fil i en bash-terminal med kommandoen:

```
$ docker compose -f docker-compose-aux.yml up -d
```

Herefter er der to muligheder til at fylde vaulten:

1. Tjek om scriptet i `script` containeren er fuldført. Den skriver ud til konsollen, hvis der er gemt variabler i vaulten.

Hvis den melder følgende fejl:

```
failed to create client: parse "http://jacob:8200\r": net/url: invalid control
character in URL
```

Skal du følge det næste skridt.

2. Gå ind i `vault-dev`-containeren, og derefter ind i terminalen. Skriv først følgende kommando, for at export VAULT_ADDR:

```
$ export VAULT_ADDR='http://0.0.0.0:8200'
```

Skriv herefter følgende kommando, for at fylde vaulten med secrets:

```
$ sh fill.sh
```

Tjek herefter om `rabbitmq` containeren er oppe at køre. Hvis ikke, så skal den startes igen.

Kør derefter `docker-compose-services.yml` i terminalen med kommandoen:

```
$ docker compose -f docker-compose-services.yml up -d
```

Nu burde alle services være oppe at køre (undtagen `script`-containeren), og vi kan nu tilgå de forskellige API-endpoints.

3. Opret Bruger

> Du kan følge med i **MongoDB Compass** for at se, om ting bliver gemt, ved at connecte til databasen i Compass: `mongodb://admin:1234@localhost:27018/?authSource=admin`

Du kan selv vælge om du vil tilgå de forskellige endpoints via Postman eller cURL scripts. I guiden benytter vi cURL scripts for hurtigere adgang og eksekvering.

> Du kan også få adgang til vores workspace i Postman med dette link, hvis du vil tilgå alle endpoints: https://app.getpostman.com/join-team?invite_code=66c0fe8b87be0d2e4a8d9095776f6fc0&target_code=92e0ede8725339269ba758278c79a4b3

For at oprette en bruger, tilgår vi `Users-service`. Du kan skrive følgende cURL-kommando ind i en shell terminal, for at oprette brugeren:

```
curl --request POST \
  --url http://localhost:4000/Users/addUser \
  --header 'Content-Type: application/json' \
  --data '{
    "FirstName": "Henrik",
    "LastName": "Jensen",
    "Address": "Sønderhøj 30",
    "Phone": "12341234",
```

```
    "Email": "henrikjensen@gmail.com",
    "Password": "password",
    "Verified": true,
    "Rating": 9.0,
    "Username": "henrik"
}'
```

> Husk at gemme dit **userID** eller find det i databasen.

Vi gemmer **userID** i en bash-variabel:

```
$ userId="<indsæt userID her>
```

Test om variablen er gemt ved at køre kommandoen:

```
$ echo $userId
```

4. Login

For at login, skal du bruge dit `Username` og `Password` til at tilgå login endpointen, som ligger i `Auth-service`. Kør følgende kommando i terminalen:

```
curl --request POST \
  --url http://localhost:4000/AuthService/login \
  --header 'Content-Type: application/json' \
  --data ' {
    "Username": "henrik",
    "Password": "password"
}'
```

Efter login, får du en JWT-token returneret. Din token ser nogenlunde ud som denne:

```
{
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9y
Zy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6ImhlbnJpayIsImV4cCI6MT
Y4NTQ0MTcyMSwiaXNzIjoiSkFEQURBRERBQURBIiwiYXVkIjoiaHR0cDovL2xvY2FsaG9zdCJ9.PKotjwX
3xb6NOilLOWtqObw6hd7WJsmYFuuIY9NyrZo"
}
```

Denne token skal gemmes(kun strengen, uden " "), da den bruges til at tilgå endpoints som er beskyttet og som kræver **Authorization** for at tilgå dem.

Vi gemmer også **token** i en bash-variabel:

```
$ token="<indsæt token her>"
```

Du kan teste om din token virker, ved at køre følgende kommando i terminalen:

```
curl --request GET \
  --url http://localhost:4000/Users/getAllUsers \
  --header "Authorization: Bearer $token"
```

5. Opret Auctionhouse

Næste trin er at oprette et auktionshus. Vi kalder `addAuctionHouse` endpointet i `Users-service` med følgende cURL kommando:

```
curl --location 'http://localhost:4000/Users/addAuctionhouse/' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $token" \
--data '{
    "Name": "G&O",
    "Address": "Groennegade 45",
    "CvrNumber": "10150817"
}'
```

> Husk at gemme **auctionID** som bliver returneret i terminalen eller find det i databasen.

Gemmer **auctionID** i en bash variabel:

```
$ auctionhousedId="<indsæt auctionId her>
```

6. Opret Article

Næste trin er at oprette en effekt, som skal lægges op til auktion. Vi kalder `addArticle` endpointet i `Article-service` med følgende cURL kommando:

```
curl --request POST \
  --url http://localhost:4000/ArticleService/addArticle \
  --header "Authorization: Bearer $token" \
  --header 'Content-Type: application/json' \
  --data '{
  "Name": "Spisestol",
  "NoReserve": true,
  "EstimatedPrice": 4556,
```

```
  "Description": "Laekker kvalitet, aegte gedeskind",
  "Category": "CH",
  "Sold": false,
  "AuctionhouseID": "'"$auctionhouseId"'",
  "SellerID": "'"$userId"'",
  "MinPrice": 2400,
  "BuyerID": ""
}'
```

> Husk at gemme den returnerede **articleID** som bliver returneret i terminalen eller find det i databasen.

Gemmer **articleID** i en bash variabel:

```
$ articleId="<indsæt articleId her>"
```

Vi kan også vedhæfte et billede til vores nye effekt ved at køre denne kommando:

```
curl --location --request PUT
"http://localhost:4000/ArticleService/addArticleImage/$articleId" \
--header "Authorization: Bearer $token" \
--form "=@$(pwd)/spisestol.png"
```

> For at ovenstående kommando virker med `$pwd`, skal man stå i mappen hvor billedet er gemt.

7. Opret Auction

Næste skridt er, at oprette en auktion med den nyoprettede effekt. Vi kalder `addAuction` endpointet i `AuctionPlanning-service` med følgende cURL kommando:

```
curl --location 'http://localhost:4000/AuctionPlanning/addAuction' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $token" \
--data '{
  "StartDate": "2023-05-30T12:00:00",
  "EndDate": "2023-06-20T14:00:00",
  "ArticleID": "'"$articleId"'"
}'
```

> Husk at gemme den returnerede **auctionID** som bliver returneret i terminalen eller find det i databasen.

Vi gemmer også **auctionID** i en bash variabel:

```
$ auctionId="<indsæt auctionID her>"
```

8. Add Bid

Til sidst skal vi tilføje et bud til vores nye auktion. Det gør vi ved at kalde addBid endpointet i Auction-service med følgende cUrl kommando:

```
curl --location --request PUT 'http://localhost:4000/AuctionService/addBid' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $token" \
--data '{
    "Price": 2400,
    "BidderID": "'"$userId"'",
    "AuctionID": "'"$auctionId"'"
}'
```

Vi har nu været igennem hele flowet for en auktion.