FullStack application for a boat auction

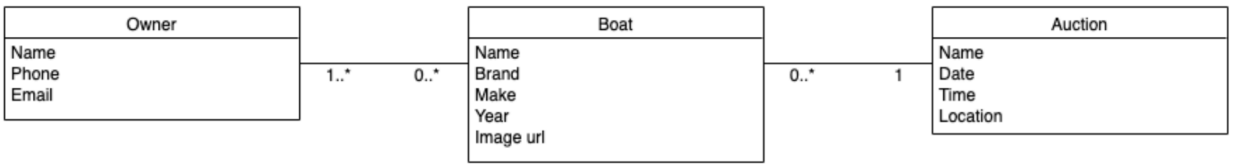
You are required to create an auction system web application, so users can log in and see what auctions are planned and what boats are being sold. Owners can update information about their boat and admins can create, update and delete auctions, boats and owners.

Your task is to create a proof of concept solution for this web-app. As of now, the requirements are a bit vague, with only the information given below. You are free to add whatever you find relevant if it makes sense for the overall task and you can provide arguments for what you did since you don't have a product owner to consult for this exercise

Non-functional requirements

- It must be implemented as a modern Single Page Application
- The backend must be implemented with Java, JPA, REST (with JAX-RS) and a MySQL database
- We do expect Test Cases for most of what you do on the backend, both UnitTests and Integration Tests (testing your REST API)
- The frontend must be implemented with React
- The project must use a modern DevOps pipeline, using “github actions” to provide a build server. This should be the VERY FIRST thing you set up, and we expect it to build, run your tests and deploy to your droplet which should be set up with your own domain name and use HTTPS.
- We expect that you can demonstrate your project, both locally (so we can add changes) and remotely on your Droplet.
- The “product owner” has come up with this initial domain model. Feel free to **add** missing fields and also **change it**, as long as you can provide arguments for why you made the changes.
- Setup test data in the database any way you like

Domain model



In the above domain model, there are 3 entities but the relationship between owner and boat is a many-to-many relationship

Functional requirements

Read all the requirements before you start coding. You do not have to implement them in the given order, if a different order makes more sense to you and if you may be pressed for time.

- The site requires the users to authenticate for all operations and only an admin can update and create new auctions. Users and roles are not shown in the model above. If you have a ready to use start project with support for this, feel free to use it for the application.

- For the client application, we would like the following features written up as user stories here:

US-1	As **a user** I would like to see all auctions
US-2	As **an owner** I would like to see my boats
US-3	As **an owner** I would like to add a new boat
US-4	As **an owner** I would like to update an existing boat
US-5	As **an admin** I would like to create a new auction
US-6	As **an admin** I would like to update all information about a boat, its owner, and the auction
US-7	As **an admin** I would like to remove a boat from an auction

- Add any features you find relevant (to showcase your skills) if they are not present in the above list.

Hints for this exercise

Make sure, in your solution, to include both backend (with tests) and frontend code. **This is much better than having a complete backend, but no frontend at all**.

Use of Github during the exam

Please make your repositories private during the 24 hours, and make them public again after handing in Friday morning. In this way the examinator and censor can view your code before the oral exam, and no one but yourself can see it during the exam.

Expected details for the hand-in

You must hand in a single document via Wiseflow with the following information, no later than 08.30 AM, the day after you received the exercise.

- Your full name and student-number
- A link to your GitHub repo(s)
- A link to your Deployed Client Application
- A link to your Deployed backend server

This document should also include a short description (5-10 lines) describing how far you got with the exercise. This can be in Danish or English. You should include a list with a few test-users found in your database including their role(s), username and password.

Important:
You **may NOT push to your GitHub repo after 08.30 AM, 24 hours after you received the exercise**. This could make your hand-in invalid. At the exam, we will probably request minor changes, and verify whether Github Actions deployed those changes to your Droplet.