Backend Design/Architecture Home Assignment

In this document we will present some conceptual challenges we encounter while working in a microservices environment. The purpose of this exercise is to test architectural perspectives and aspects together with basic hands on for a working system.

Guidelines

- Please note - usually there is no "right answer" for each problem, every solution choice has its pros and cons, and the architecture context might affect on choosing the solution, we would like to **Discuss** the options in those scenarios at a high level.

- We estimate that overall (both challenges) should take between 3-4 hours

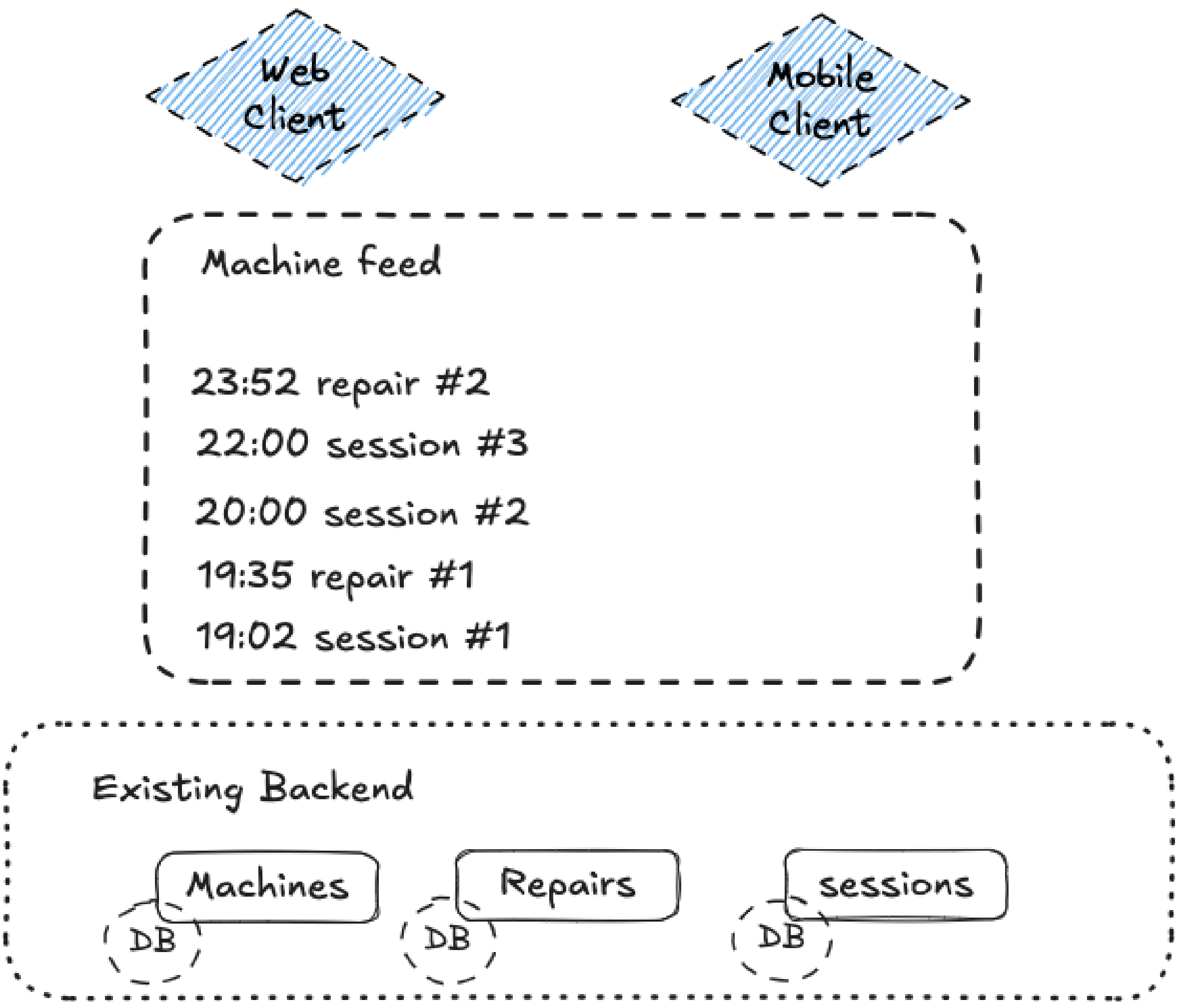
Challenge Requirements

As you probably know, Augury is providing an end to end solution around Predictive maintenance to industrial machines, therefore our main business entity is a Machine.
This design will be the base for our frontal technical interview by simulating a Code Review & Design Review which are ingrained in Augury R&D culture.

The new product requirement

1. Build a UI page both for mobile and the web clients containing a grid (table) to present "Machine feeds".
2. "Machine feeds" are activities related to a machine that took place on a certain machine over time. (the machine details entity is called "machine-configuration")
3. We have 2 types of machine-feed activities that can be related to a machine
    a. Repairs - descriptive information (text and sometimes image) to explain about a repair (physical fix) performed on a machine (created by a user (dedicated UI - out of scope))
    b. Session (machine recording) (created by our IoT devices every 1 hour)

This is the existing architecture:



Technical guidance

1. The all 3 entities of machine (machine-configuration / repairs / sessions) have very basic DB schema (you can extend if needed)
    a. Id
    b. Created_at
    c. Data (unstructured)
2. All 3 entities have CRUD operations (& UI if needed) implemented
3. The URL path will include machine_id that will be your correlation to all activities. app.augury.com/machine-feeds/:machineId

Your mission, should you choose to accept it…

Output #1 - Working "basic" system

1. In this output, we would like to validate your ability to provide a working solution in a reasonable time frame.

2. Implement the **basic microservices** required for this task with the **API** required by the frontend to fill the grid data
    a. You can choose your preferred Backend & framework (we use Golang & Gin)
    b. Please focus on the **main** functionality only required for the task
    c. Our system architecture in Augury is based on **distributed microservices** approach, we expect your design to be the same.
    d. You can choose to implement the **Naive** approach and be ready to discuss/present more options in your diagrams for improvements
    e. **Ignore** the Frontend part from the requirements (make sure to implement the api)
    f. **Ignore** authentication / authorization
    g. **Ignore** CRUD operations for the entities (machine-configuration , repairs, sessions), you can assume they are already available.
    h. please mock the DB data with a json format like the example below
3. Share the results in zip file / github link with run.info guidelines
4. Be ready for "code review" and "design review" discussions with the hiring team

Output #2 - HL schematic architecture diagram

1. Share a schematic architecture diagram (you can use excalidraw.com or any other favorite tool) that can explain the main design components from UI to the Backend. (we value simplicity, "rectangles" and "lines" can do the work :))

Output #3 - HL Design review F2F discussion

1. Be ready for a **F2F** discussion about your current implementation and other more robust options
2. (**optional**) 1 Page Design review document (word / google docs format) that can explain your thoughts and potential solution for the product requirement.
3. Provide 2 optional alternatives with pros/cons for your decision.
4. Be ready to describe the following in details
    a. Main API's & their parameters
    b. Main DB entities schema
5. Take into consideration for a discussion (**no need to implement**)
    a. Document any important architectural decision you take
    b. Communication channels (sync/async)
    c. Performance & Scalability considerations
    d. Cloud deployment best practices
    e. "Taking the feature to production"

Good Luck