2. Exercise – Battleship Microservices with REST IPC

Analyze your implementation of the Battleship application and divide it into multiple microservices. Try to find independent functionalities to split the application. Implement inter service communication using REST APIs.

Try to have some domain logic in each microservice if possible. Try to organize your domain models according to DDD aggregates and use primary keys (ids) for referencing entities via REST APIs.

Requirements:

- Your architecture should at least consist of three individual microservices (three Spring Boot projects)
- Remember that each microservices has its own persistence (database)
- **A user should not interact with individual microservices. Keep the openapi Dashboard from Exercise 1 and use inter service communication to maintain the same functionality as in Exercise 1.**

Make sure that each microservices has a unique port in the application.properties/application.yaml.

2a. Exercise – Add circuit breakers to your project

Implementing circuit breakers in your project ensures resilience and stability by preventing cascading failures in distributed systems.
Add circuit breakers to your project as discussed in the lecture. There is a reference implementation in ilias.

Follow Spring Boots approach regarding code structure (controllers, services, repositories and entities).
Document your APIs using openapi as mentioned in the lecture. (See https://springdoc.org/)
The path to access openapi will be: http://localhost:8080/swagger-ui/index.html

Submission:

Hand in your solution in form of a .zip archive including your Spring Boot projects. Make sure they are executable. Additional documentation should be added to a README.md placed in the root directory of each project.