Technical Documentation: survival-helper

Contents

Introduction

survival-helper is a service created to support the existence of human race, according to the requirements provided by a the IOCO team. The main functionalities of the service are,

1. Add a survivor
2. Get Survivors
3. Update Location of a survivor
4. Update health status of the survivor
5. Get a report of statistics of the survivors
6. Get a List of robots from an external service: Robot CPU

Decisions

Since there is no technical specification available to decide between database technologies, h2 in-memory database has been used. Unit testing has been only covered for specific classes which have considerable amount of logic. Get Survivors API added, which was not on the specification.

Motivation

When designing the solution, following key points were considered.

1. The application should be well structured to accommodate future changes, and the coupling between each layer should be minimized.
2. Cognitive complexity of the code should be minimized for anyone to understand quickly.
3. Service classes should be well covered with unit tests, and no code smells or vulnerabilities.
4. Http calls to the other services should be non-blocking to preserve reactive nature.

Architecture

The architecture of the application is designed according to the concepts of **hexagonal architecture**, which uses **ports and adapters** to segregate between layers. Furthermore, it ensures **dependency inversion**.

The application has not segregated into modules based on layers, since the requirement is not very complicated, but the separation of concerns has done using packages instead.

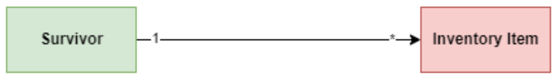
The application has three main packages: **application, domain, provider, and persistence**, which represent each three main service layers.

The **application** package consists of **Controllers**, which holds the gate to enter the domain layer. The **domain** package is very important, since it contains business logic, carefully separated from others. The **persistence** layer supposed to handle all the functionalities related to the database operations, and **provider** handles the communication between the survival-helper and the external services

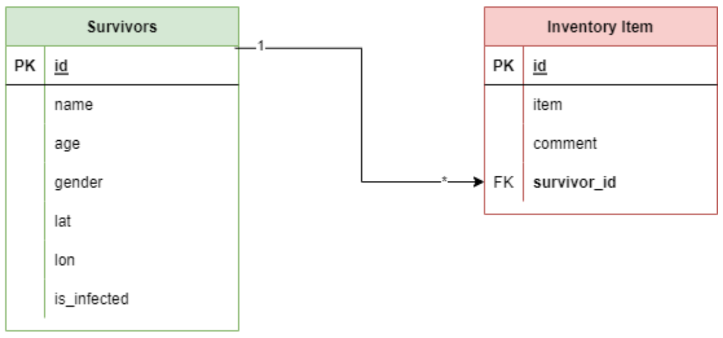
Entity Relations

There are two entities found, Survivor and the Inventory of items they have. Survivor entity connects with Inventory Items entity in **one-to-many** relationship from the survivor’s perspective.

High-level ERD



Extended ERD



Class Locations

1. The location of **Service class**:
   main -> java -> *com.ioco.survivalhelper -> domain -> services -> SurvivorHandler.java*

2. The location of corresponding **Unit test classes**:
   test -> java -> *com.ioco.survivalhelper -> domain -> services -> SurvivalHandlerTest.java*

   - *Corresponding **provider** unit tests:* test -> java -> *com.ioco.survivalhelper -> persistence-> adapters -> SurvivorPersistenceAdapterTest.java*

Frameworks and plugins

Spring Boot 2.7.2 has used as the java framework to develop this API. Most of Spring Boot features (AOP, IOC) have been used to develop the application. Lombok plugin has used to enhance code readability, and Junit 5 Jupiter is used as unit testing framework. Since we have not connected the application to any databases, In-memory database H2 was used with JPA.

Code Quality

The static code analysis was performed on code base by using Sonar Lint plugin.

Prerequisites

1. JDK version 11
2. Gradle 7.5
3. Postman – optional

How to test?

All the endpoints have been defined on the postman collection included in root directory, and API documentation has been published by using postman.

How to use the Collection

- Import the postman collection into your postman
- Change baseUrl variable accordingly
- Firstly, add some survivor details by using Add survivor details API (Note: Survivor Id should be a valid UUID, which can be generated using https://www.uuidgenerator.net/)
- Then you can try all the update APIs and get report API as well.
- Get Robots API is independent from Survivor details, and you can try this anytime, but it needs internet connection to retrieve robot list from the external service.

API Documentation

Please access the Postman generated API documentation from here.

https://documenter.getpostman.com/view/18834681/2s7YYoBSK9