Inatel

Practical Project Evaluation (Java Developers)

These evaluation goals encompass the validation of both theoretical and practical technical knowledge. Primarily, they involve comprehending the given business and technical requirements for a sample application. Subsequently, the developer is tasked with utilizing key technologies and concepts commonly employed within the Inatel project's environment to construct the application.

Installation

The following software are mandatory to be installed.

Mandatory Software	Link to download
Docker	https://docs.docker.com/docker-for-windows/install/
Java JDK 17 or later	https://jdk.java.net/archive/
Maven	https://maven.apache.org/download.cgi

The following software are optional and can be replaced by others for the same purpose.

Optional Software	Link to download
Eclipse IDE for Enterprise Java and Web Developers	eclipse.org/downloads/packages/release/2022-09/r/eclipse-ide-enterprise-java-and-web-developers
Postman	https://www.getpostman.com/downloads/

General information

Internet can be used with no restriction.

The project with all the source code produced must be delivered at the end of the test.

Configuration

The application to be developed will use two services that run on Docker containers.

You will need two images to run the services below which also will be consumed by application to be developed (more information will come later).

- MySQL: https://hub.docker.com/_/mysql
- publisher-manager: https://hub.docker.com/repository/docker/adautomendes/publisher-manager

Create a new network to be used for all services you need:

```
docker network create inatel
```

The first service is a MySQL database service. To start it, run the following command line at Windows Power Shell.

```
docker container run --name mysql --network=inatel -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=bootdb -p 3306:3306 -p 33060:33060 -d mysql
```

The second service is a REST application that will be consumed by application to be developed (more information will come later). To start it, run the following command line at Windows Power Shell or WSL.

```
docker container run --name publishermanager --network=inatel -p 8080:8080 -d adautomendes/publisher-manager
```

For the publisher-manager it is possible to customize the application environment which it will run setting the following env vars:

- SERVER_HOST
- SERVER_PORT
- MYSQL_HOST
- MYSQL_PORT
- SPRING_PROFILES_ACTIVE

For example:

```
docker container run --name publishermanager --network=inatel -e SERVER_HOST=localhost -e SERVER_PORT=8080 –e MYSQL_HOST=mysql –e MYSQL_PORT=3306 –e SPRING_PROFILES_ACTIVE=prod -p 8080:8080 -d adautomendes/publisher-manager
```

Any docker image can be stoped and re-started at any point in time. Any data stored on it is wiped if the container is recreated (removed and run again).

To stop a container, first issue [`docker ps`] to get container's ids. Then, issue [`docker stop {id}`]

to stop a specific container.

The following sections describes the needed information to create the application from scratch.

Each section contains a new requirement. It is recommended to implement the application in the sequence the sections are presented.

The application should preferably be developed using Maven. Use the most suitable low level

architecture for the given problem.

A. Store and read games

The application to be developed, called game-manager, is a REST based application whose purpose is to store time played from known games.

For example, a user wants to register that played **Sonic** game from **Sega** publisher at **01/12/2022** for **3 hours**.

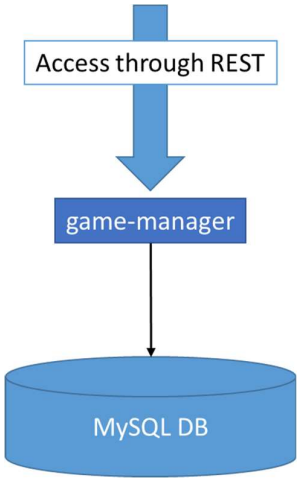
A user can register as many time played as it wants for the same game. Also, a user can register time played from different games.

Finally, a user can read games registered.

To store and read games, the application should expose REST APIs through port 8081.

The games should be stored at MySQL DB running on Docker container, described in previous section.

The following picture illustrates the application:



Game-manager application should be developed using **Spring Boot 3.0** or later.

The payload for creating or reading a game should follow the depicted example (game 'id' must be automatically generated by the application developed using UUID format).

```
{
  "id": "c01cede4-cd45-11eb-b8bc-0242ac130003",
  "publisherId": "nintendo",
  "name": "Mario",
  "timePlayed": {
    "2023-05-01": 10,
    "2023-05-02": 2,
    "2023-05-03": 3,
    "2023-05-04": 4
  }
}
```

In total, the following operations should be exposed at game service:

- Create a Game
  - POST http://<game-manager-host>:<game-manager-port>/game
- Read Games by publisherId
  - GET http://<game-manager-host>:<game-manager-port>/game?publisherId=<publisherId>
- Read all Games
  - GET http://<game-manager-host>:<game-manager-port>/game

B. Validating a publisher before processing

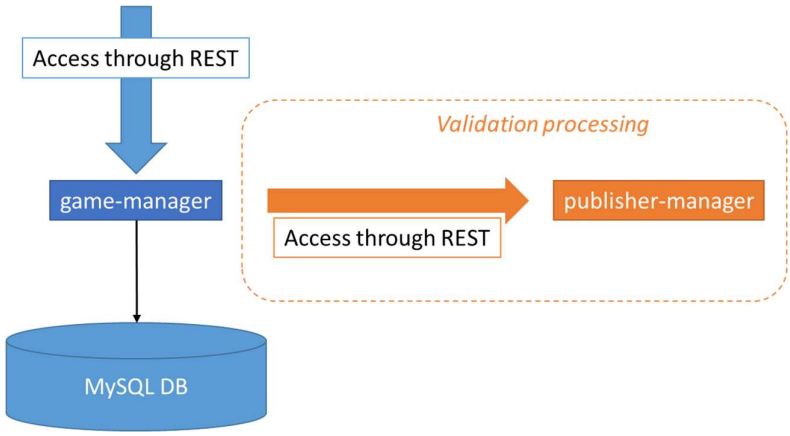
A new service is introduced to the solution in which the application being developed is part of.

This service is called publisher-manager. It is responsible for maintaining a list of publishers that can have games published at game-manager.

In other words, a user is allowed to create games on game-manager only if the publisher is registered at publisher-manager.

Hence, game-manager should be modified to verify, before creating a game, if a received publisher is already registered at publisher-manager. In case it is not, an error should be returned at create game REST API.

The following picture illustrates the application:



publisher-manager service runs on a docker container on port 8080, as described in previous section.

As mentioned in the picture, publisher-manager service exposes REST APIs to fulfill its needs.

Any outside user can register publishers on publisher-manager.

game-manager just reads registered publisher to validate the game creating operation.

- GET http://<publisher-manager-host>:<publisher-manager-port>/publisher

  Get the list of publishers registered. By default, when server is started, publishers NINTENDO and SEGA are already registered.

- POST http://<publisher-manager-host>:<publisher-manager-port>/publisher

```
{
  "id": "konami",
  "name": " Konami Holdings Corporation "
}
```

  Register new Publisher.

C. Caching registered publishers

To avoid accessing publisher-manager service at each game creating operation, game-manager application should cache locally (in memory) the registered publishers.

When validating a game, whenever the cache is empty, it should populate it from publisher-manager service.

If cache is not empty, it should validate with cached data.

The cache should always be cleaned when a new publisher is registered at publisher-manager.

To achieve this, two new functionalities should be introduced.

game-manager should register itself at publisher-manager during startup. To register itself, the following publisher-manager service must be invoked.

- POST http://<publisher-manager-host>:<publisher-manager-port>/notification

The body of the service should follow the below example. It should contain the host and the port of the application being registered (in our case, game-manager).

```
{
  "host": "localhost",
  "port": 8081
}
```

Whenever publisher-manager receives a request to register a new publisher, it tries to invoke a notification service at all registered applications.

Hence, game-manager should expose a new REST service to accept notifications from publisher-manager.

publisher-manager expects the following URL.

- DELETE http://<game-manager-host>:<game-manager-port>/publishercache

Publisher-manager replaces <game-manager-host> and <game-manager-port> by the proper data from registered applications.

Game-manager, upon reception of this notification, should clean the cache.

D. Testing

Create unit tests for any developed service class.

E. Additional requirements

Create a Docker image (Dockerfile and docker-compose) for game-manager, so that it can run in a container.

Deliverables

The project must be delivered as a **PRIVATE** GitHub repository link sent by email to:

- Fabíola Prado: fabiola.prado@inatel.br
- Adauto Mendes: adauto.mendes@inatel.br

Also, add the following reviewers as collaborators in repository:

- adautomendes

Important: projects delivered in public repositories will be automatically disqualified.