REST API FOR SAAS BOOKING PLATFORM

Tomislav Landeka (tomo.landeka02@gmail.com**)**
https://www.linkedin.com/in/tlandeka/
https://github.com/tlandeka/saas-booking-platform

Foreword

This document describes the application design that I have made. Before You start the read I would like to mention that I do not have for example a 100% test coverage, but I presented the way I write tests. These are the features that have not finished completely(because I wanted to spend more time on the architecture), but I have done it partially which means that I have shown the way I do it:

- I do not have 100% test coverage
- I did not validate **all** client inputs
- The API uses only IllegalArgumentException and NofFoundException
- I have not used any Logger, but it is easy to add Log4j2 or Logback

Task Description

Create a saas platform for boutiques, studios, and gyms which allows business owners to manage their courses, classes, members,memberships etc,.

Story - Create Classes

As a studio owner I want to create classes for my studio so that my members can attend classes

Acceptance Criteria

- Implement an API to create classes(`/classes`). Assume this api don't need to have any authentication to start with.
- Few bare minimum details we need to create classes are - class name, start_date, end_date, capacity. For now, assume that there will be only one class per given day. Ex: If a class by name *pilates* starts on 1st Dec and ends on 20th Dec, with capacity 10, that means *Pilates* have 20 classes and for each class the maximum capacity of attendance is 10.
- Use MySQL relational database storage to save state.
- Use Restful standards and create the api end point with proper success and error responses.

Story - Book for a class

As a member of a studio, I can book for a class, so that i can attend a class.

Acceptance Criteria

- Implement an API end point (`/bookings`). Assume this api don't need to have any authentication to start with.
- Few bare minimum details we need for reserving a class are - name(name of the member who is booking the class), date(date for which the member want to book a class)
- Use MySQL relational database storage to save state.
- Use REST standards and think through basic api responses for success and failure.
- Consider the scenario of overbooking for a given date. Ex: 14th Dec having a capacity of 20 and number of booking cannot be greater than 20.

Implementation

While I was working on implementation I followed Domain Driven Design(DDD) approach. The DDD follows the best practices, SoC, SOLID principles, decoupling the dependencies, etc.. DDD is too much for this application but I just want to show how to use DDD.

When we talk about technologies, this is a list of used technologies:

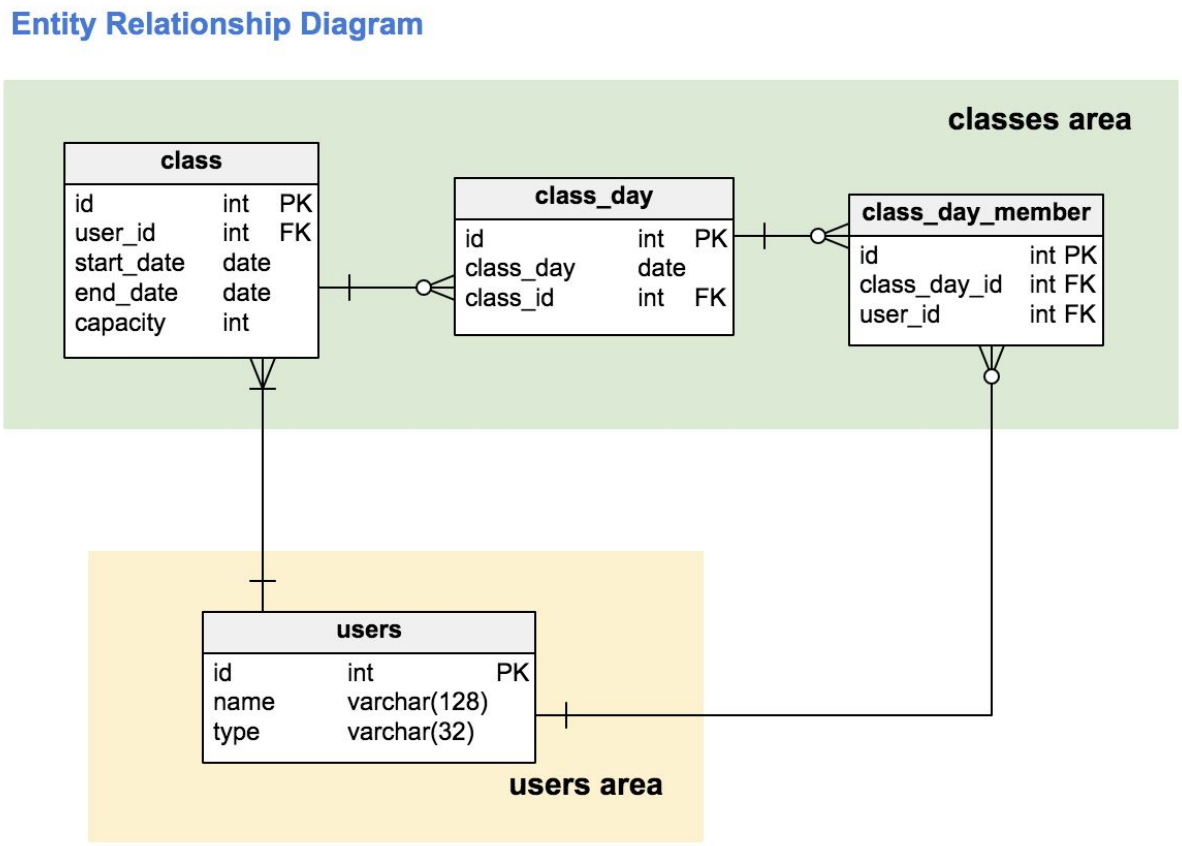
- Java 8, Maven, Spring Boot (SB), SB JPA, SB centralized error handling, SB auto-configured beans
- Design Patterns, DDD, Aggregate Root, Unit/Integrations/e2e testing
- Docker, Docker Compose
- MySQL and Vertabelo(database designer)
-

When we talk about tests, this is a test coverage statistics:

Element	Class %	Method %	Line %
application	100% (10/10)	86% (32/37)	87% (70/80)
domain	85% (6/7)	76% (40/52)	88% (119/135)
infrastructure	83% (5/6)	36% (7/19)	41% (17/41)
ExcerciseApplication	100% (1/1)	0% (0/1)	33% (1/3)

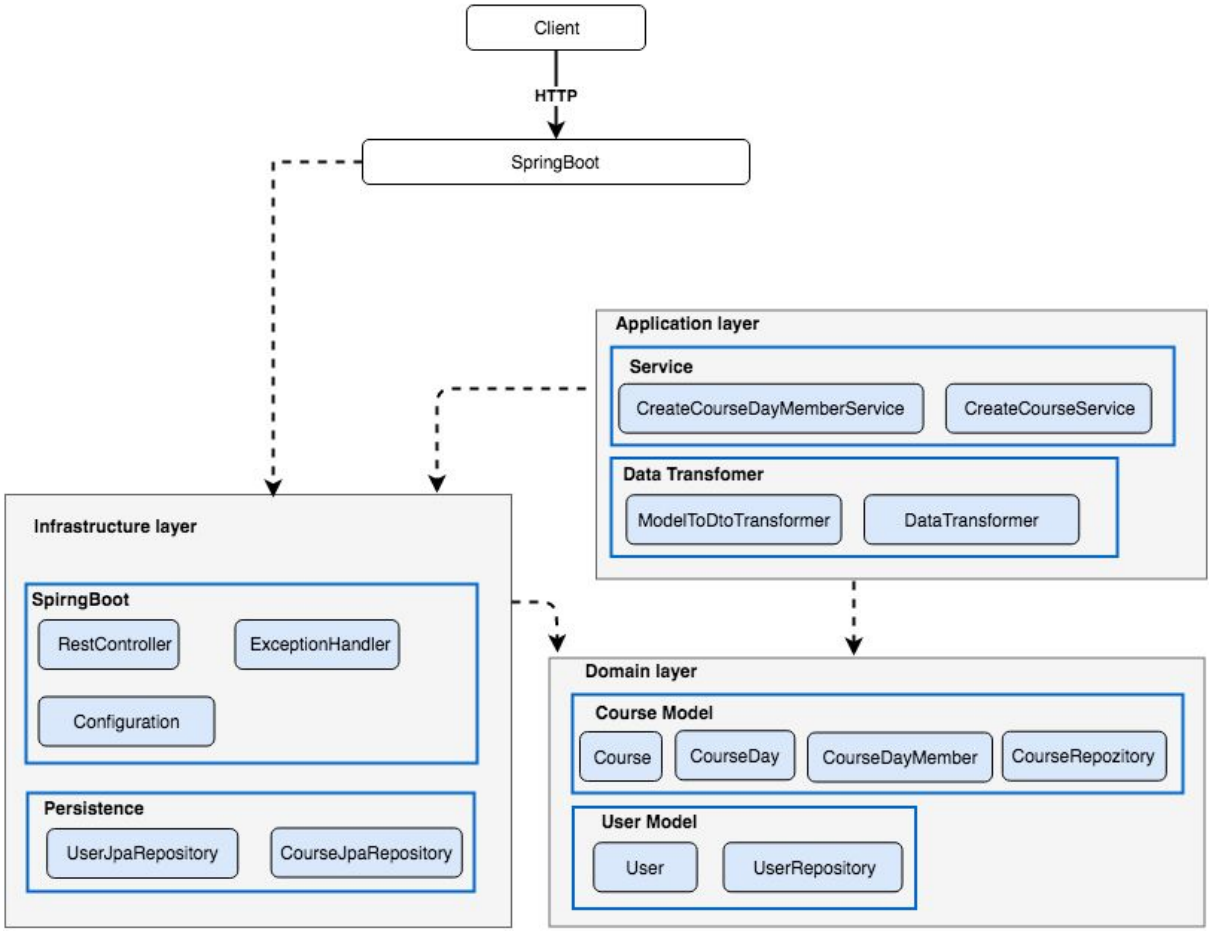
Database ER diagram

I have made an application that can create one class or booking and store it to the database but with a condition that user details already exist in the database(Let's assume that someone already added the users into our saas booking platform) therefore I have made database migration that inserts the test “users” data into database.
According to that I have created the database ER diagram in this way:



This diagram tells that one *user*(with type=OWNER) can have multiple classes, and one *class* can have multiple *class days*(it depends on start_date and end_date in the class table). Also, the diagram tells that one *class day* can have multiple class day member(a member is user with type=CLIENT)
The table **flyway_schema_history** is auto-generated table by the migration tool.

Architecture Diagram



Source code
The source code of the application is available on this link:
https://github.com/tlandeka/saas-booking-platform.

Take a look on the *README.md* file in order to run the application.

Basic API usage

The API provides two actions in order to manage create and book course/class.

Create class action

Method: **POST**
Route: /classes
Request body example:

```
{
      "userId": 5,
      "name": "Pilates",
      "startDate": "2019-01-01",
      "endDate": "2019-01-10",
      "capacity": 5
}
```

Response body example:

```
{
    "id": 111, //ID of created class
    "name": "Pilates"
}
```

Book class action

Method: **POST**
Route: /bookings
Request body example:

```
{
      "userId": 6,
      "courseId": 111,
      "courseDate": "2019-01-06"
}
```

Response body example:

```
{
    "id": 222, //ID of booked class
    "name": "Pilates"
}
```