Project Documentation

February 24, 2023

1 Design

The application is structured in layers. Each layer has an area of concern for which it should be responsible, with as little mixing as possible between the layers. Furthermore, any one layer should only touch the layer immediately beneath it, and never any deeper than a single layer. The layer hierarchy is as follows:

- Controllers
  - Services
    - Models
    - Repositories

1.1 Controllers

Controllers are responsible for receiving Http Requests and creating and sending Http Responses. The pattern is simple:

1. Get a request object
2. Authenticate the user (via the `UserService`)
3. Retrieve the contents of the response from the service layer
4. Send the response

1.2 Services

Services are responsible for all the logic that happens in the application. Methods are provided for i.e. authentication, trading, and battling. The pattern here is more variable but follows this general list of objectives:

- Sanity checking of data
- Usage of repository methods for input/output from the database
- Performance of some kind of operation on the data such as calculating a user's stats
- Returning valid data

1.3 Models

Models define the actual "objects" involved in the application. These can be more or less concrete, but they serve a couple of purposes:

- Structuring data
- Defining objects

1.4 Repositories

Repositories depend on the models for defining what the database schema should look like. The repository then provides simplistic database IO, including creating, reading, updating, and deleting data. (CRUD)

- Defines schema
- Provides input/output
- Data validation, in the sense that only data which conforms to the models can be read/inserted/etc.

2 Lessons Learned

This project was an excellent learning experience. I now feel confident in my understanding of two important concepts in the world of software engineering:

1. REpresentational State Transfer (REST)
2. Model View Controller architecture

Learning these topics has already improved my skills in a professional context.

As time went on during the development of this project, I realised again and again that I had made mistakes or suboptimal architectural decisions in previous parts of the application. This should be apparent to anyone looking at the code in the order I wrote it: Users -> Cards -> Trading -> Battles

I had to prevent myself from going back and trying to fix those issues in order to finish the project on time. I would like to rewrite this entire project (potentially in another language) someday with two goals: Cementing the lessons learned, and improving the architecture from the ground up.

Rewriting it in another language is no comment on the effectiveness of Java; this application is simply, in general, large enough to make acquainting myself with more of the features of another useful language (like PHP) a neat sub-goal of rewriting it.

Something that I found sorely lacking while developing this application was regular code reviews with more experienced colleagues. I'm certain that much of the code I wrote has potential to be broken in ways I never even considered, or that it doesn't conform to accepted Java "best practices," or that it's wildly inefficient, or any number of other problems. This is not an excuse for poor code quality, but it is food for thought.

One other note: I found writing meaningful unit tests to be the most confusing part of the development. This is an area where I need to improve.

3 Time Spent

Note: This information is only approximate; there were times when I worked on it without recording the time spent, and there were times where the clock was running while I took a break, etc.

Furthermore, for simplicity, I didn't split up the time spent working on Controllers and Services, or from one Controller to another. I decided that constantly worrying about the running clock and which context it was in wasn't worth the mental effort.

I categorized the Controllers, Services, Repositories, and Models into roughly four areas according to the OpenAPI specification:

1. Users

2. Cards

3. Trading

4. Battle

I wrote them in this order, and mostly worked on the Repositories and Models first, then the Controllers, and finally the Services.

I spent more time (on the order of maybe four hours) on the Database, Synchronization, and Unit Test categories than is recorded here in particular, as I often needed to research these topics in order to understand certain implementation details and strategies.

Table 1: Clock summary at *[2023-02-24 Fri 13:37]*

Headline	Time	
Total time	**2d 3:08**	
Monster Card Game	2d 3:08	
Database [4/4]		5:59
Controllers [4/4]		1d 12:29
Unique Feature [1/1]		1:10
Synchronization [2/2]		2:13
Learn to use Unit Tests [1/1]		4:10
Documentation [4/4]		1:07

All of this information was generated from the project progress tracking file, mtcg.org The git history of that file provides a nice overview of how development was structured and accomplished.

4 Link to git

MTCG