Monster Trading Cards Game

Design

Program Design

I decided to follow a modular design architecture and separating concerns as far as possible to make the program more maintainable and testable.

The Program is designed as follows:

Program

The starting point of the application, responsible for initializing and starting the server and setting up the necessary configurations as well as the database.  
  
Server

- The starting point of the application, responsible for initializing and starting the server and setting up the necessary configurations.

Middleware

Act as filters that process incoming requests before they reach the controllers. Currently there is only one middleware implemented (“AuthenticationMiddleware”), but due to the modular implementation, additional middlewares can be added really quick.  
The middleware also extracts the user and stores them in the current request to be used by the corresponding controllers.  
  
Router

The Router is responsible for routing the request to the corresponding controller method using Attributes and Reflections. When starting the server, the Router detects available route methods with their corresponding attributes and maps them to their corresponding controller function. Once a matching route is found, an action (a delegate to the controller method) is executed with the context of the HTTP request, including any extracted parameters.

Controllers

The controllers are the entry point of the mapped request. They handle the business logic by making use of the corresponding repositories. They directly return responses to the client.

Repositories

The repositories are responsible for abstracting the data layer from the business logic.  
They act as an intermediary between the application's business logic (handled by Controllers) and the data source (the database).

All Repositories inherit the database connection from the BaseRepository, enabling them to access and manipulate the database consistently and securely. This shared inheritance ensures that each repository can perform data operations efficiently.

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte BeschreibungDatabase Design

Lessons Learned

This project played a main role in my programming journey.

Benefits of modular design and separation of concerns

At the beginning of the project, I did not pay too much attention about loose coupling of the components and separation of concerns which worked fine at first. But when I started testing, I noticed how hard it was to separate functionality which I wanted to test. So I made a decision and refactored the whole router functionality to be less dependent on

The important role of testing

I learned how important testing during development really is. At first, I wrote several parts of the program without testing which worked fine at the time. But when I started to write some tests, I quickly found some errors in my code or missing logic which I did not notice before. After this experience, I immediately wrote tests whenever I implemented new functionality. This positively impacted the quality of code I wrote.

Testing

Unit tests

I focused on testing the main functionality of the program. I mainly wrote unit tests for the controllers, repositories and services to guarantee that the core functionality and logic is working as intended.

Some unit tests, although not intended to, are implemented in form of integration tests. For example, to test the functionality of user creation, the test needs to access the database which makes it a integration test. (I think?)

Integration test

For integration test, I used the provided curl script.

Unique features

Feature 1: Trade Random Card for Coins

This feature allows a user to trade a random card from their stack (excluding cards in their deck) for coins. In this way the user is able to acquire coins.

The user's stack is converted to a list, and a check is performed to ensure there is at least one card available to trade. A secondary check ensures that there are cards in the users stack available for trading for coins. A random card not in the deck is selected for the trade.

Feature 2: Merge Cards

This feature allows a user to merge two random cards from their deck into a new card, provided they have different element types. This operation costs the user 3 coins. Two random cards are selected from the user's deck. A check ensures that the user has at least 2 cards in their deck and that the two selected cards have different element types. When these requirements are met, the user can merge two cars.

Tracked time



Total time: 80,5 hours

GitHub

<https://github.com/markus-roe/MonsterTradingCardsGame/>