

MCTG - SWEN1 - SEM3

How to use the server

- git clone <https://github.com/SchaqO/MCTG.git>
- Start the programe
 - Run Main.java
 - Sever accepts up to 5 clients at the same time
- Database required (postgres only since only those drivers are bundled)

Additional resources

- CURL file
- DB ERM Diagramme
- JUnit library
- GSON library
- Postgre JDBC library
- Lombok library

Functions

- User:
 - a user is a registered player with credentials (unique username, password).
 - a user can manage his cards.
 - a card consists of: a name and multiple attributes (damage, element type).
 - a card is either a spell card or a monster card.
 - the damage of a card is constant and does not change
 - a user has multiple cards in his stack.
 - a stack is the collection of all his current cards (hint: cards can be removed by trading).
 - a user can buy cards by acquiring packages.
 - a package consists of 5 cards and can be acquired from the server by paying 5 virtual coins
 - every user has 20 coins to buy (4) packages.
 - the best 4 cards are selected by the user to be used in the deck
 - the deck is used in the battles against other players.

Spiel Anleitung

Your cards are split into 2 categories:

- monster cards
 - cards with active attacks and damage based on an element type (fire, water, normal).
 - The element type does not effect pure monster fights.
- spell cards
 - a spell card can attack with an element based spell (again fire, water, normal) which is:
 - effective (eg: water is effective against fire, so damage is doubled)
 - not effective (eg: fire is not effective against water, so damage is halved)
 - no effect (eg: normal monster vs normal spell, no change of damage, direct

comparison between damages) Effectiveness:

- water -> fire
- fire -> normal
- normal -> water

Cards are chosen randomly each round from the deck to compete (this means 1 round is a battle of 2 cards = 1 of each player). There is no attacker or defender. All parties are equal in each round. Pure monster fights are not affected by the element type. As soon as 1 spell cards is played the element type has an effect on the damage calculation of this single round. Each round the card with higher calculated damage wins. Defeated monsters/spells of the competitor are removed from the competitor's deck and are taken over in the deck of the current player (vice versa). In case of a draw of a round no action takes place (no cards are moved). Because endless loops are possible we limit the count of rounds to 100 (ELO stays unchanged in case of a draw of the full game).

Unit tests

For the unit tests I kept it simple and mostly just tested the objects and the parameters they were constructed with. Meaning I checked the a class object after construction if the correct paramters were passed during creation. Aswell as some tests to check the correct CardType and Elements That includes classes such as the User, Profile, CardElement, CardType and Holder. Having several tests within each testing class.

Lesson learned

It was definitely a massive challenge when creating the REST API without any framework, especially since learning material provided from the FH was very short and dry and by far not enough. Without any help it was a huge challenge, taking me several days and hours to grasp some conecpts. However from this challenge I believe my coding skills have improved significantly. Especially in the sense of object-oriented programming. Trying to understand the concepts of HTTP requests, methods and responses has been very crucial and informative to general knowledge in the field. Continuing with DB management, the project was almost a whole package that included many aspects of programming and would require a lot of time invested to complete properly. I massively underestimated the project, this ended up being a massive mistake, I kept pushing the deadlines until it was too late. This has resulted in me being more careful with deadlines and trying to stay on track properly from now, to not fall behind and get into the same tedious situation as this project.

Time tracked

It was hard to track time as I had many breaks inbetween, however I can say that since I started this project late into the semester, i got too close to the deadline, as close as a few days. Therefore this has led to me working most of the day across several days on this project to get it done. I would probably say close to 4-8 hours a day for several days. And I am yet to fully understand every single concept used in this projet.

GitHub Link

<https://github.com/SchaqO/MCTG>