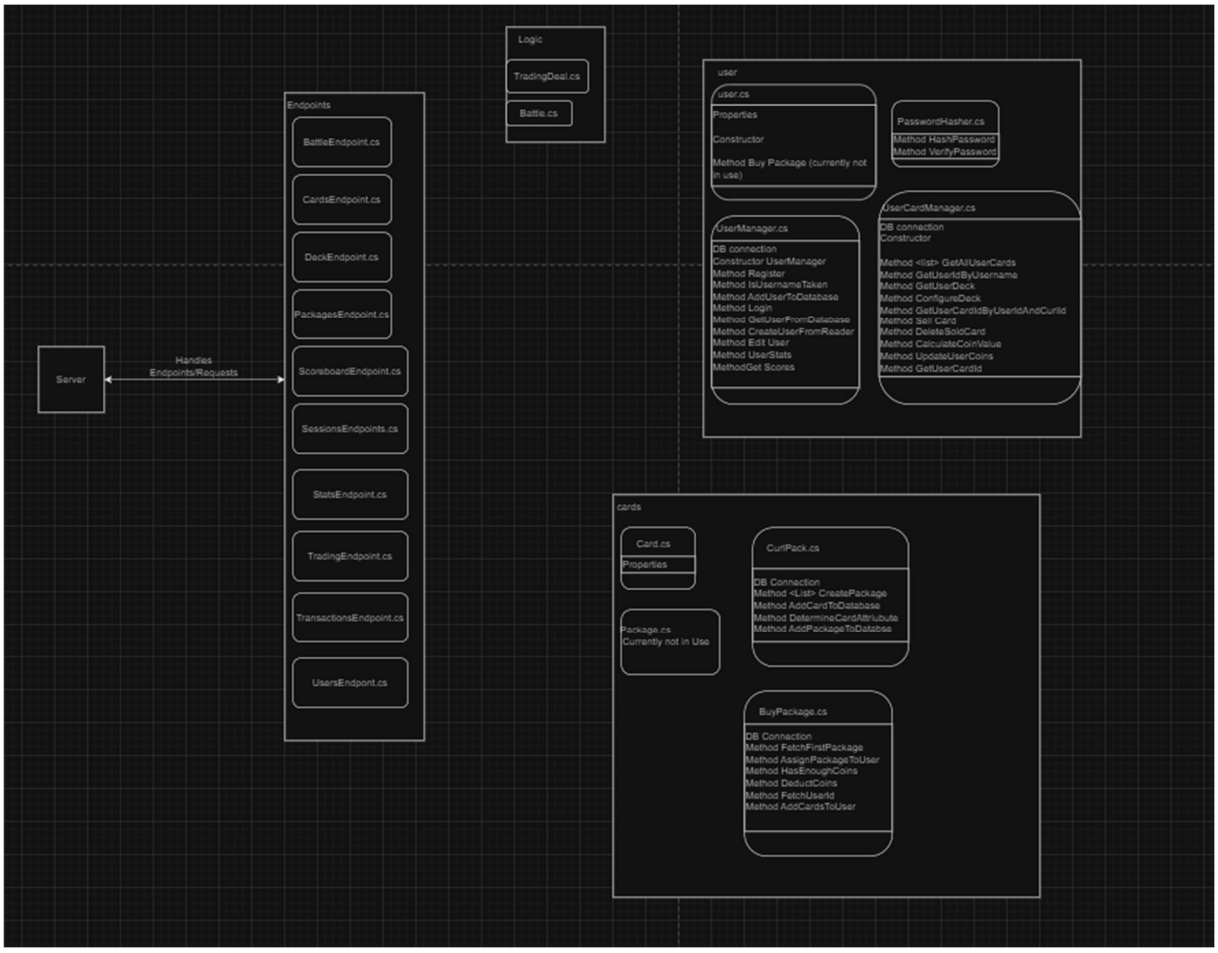
Protokoll

Design:



User werden nach der Registrierung in der DB gespeichert und das Passwort gehashed.

Registrierte User haben Stats, welche bei der Erstellung vorgegebene Standardwerte haben. Zudem haben User Karten welche ebenfalls gespeichert werden, sobald Sie diese durch ein Package erwerben. Die Karten/Packages werden vom Admin erstellt.

Jede Karte wird durch ein Package erstellt. Über diese PackageID wird dann auch das Package gekauft und die Karten dem User zugewiesen. Ein User kann sein Deck erstellen, welches aus exakt 4 Karten besteht, und tritt mit diesem Deck zum Battle an.

Server bearbeitet die Requests die reinkommen und „leite“ diese an den geforderten Endpoint weiter. Diese ruft dann die Gewünschten Funktionen/Methoden innerhalb der Klassen auf.

Unique Feature:

Die Packages werden jedes mal beim kaufen erstellt. Die Karten Attribute werden per Zufall gewürfelt. Der User hat die Möglichkeit Karten an den „Computer“ zu

verkaufen. Hier wird geschaut ob die Karte ein Element besitzt wenn ja gibt es eine münze für die Karte. Hat die Karte einen Schadenswert von 50+ dann gibt es 2 münzen und die Kombination von Element und Schaden ergibt 3 Münzen. Außerdem erhält der Gewinner eines Battles eine Münze.

Lektionen:

1. Curl Script starten und Json text auslesen und convertieren (de/serialise)
2. HTTP Requests lesen und diese an die Richtigen Endpunkte weiterleiten
3. Threading, Queue
4. Postgre Database mit SQL
5. Benutzung des Debuggers / Code Debugging
6. Unittest mit Nunit

Unit Test Design:

Wurden erstellt um auch ohne CurlScript die Funktionalitäten zu überprüfen. Ähnlicher Input wie im CurlScript. Falls nun änderungen in Funktionen stattfinden muss nicht das CurlScript durchgehend laufen um zu versichern, dass noch alles funktioniert, sondern es können die Tests durchlaufen und etwaige Fehler aufzeigen.

Getestet: Battle logic und Passworthasher. Es wurden alle Regeln für die Karten getestet da diese die Grundfunktionalität des Programms bildet. Die regeln mit dem Script zu debuggen ist fast unmöglich. Man müsste jedes mal den Ganzen script durchlaufen lassen da Karten besitzer wechseln. Außerdem wird nicht jede Karte gespielt somit kann es sein, dass manche Regeln nie zur Anwendung kommen.

PasswortHasher: Muss überprüft werden ob Passwort (richtig)gehasht wird denn PW als Text in einer DB speichern ist ein NoGo.

Unique Feature:

Gewinner bekommt nach einem Battle eine Münze. Karten können verkauft werden. Der wert der Karte wird bestimmt ob es ein Element hat (1 Münze), Dmg über 50 (1 Münze) und wenn beides übereinstimmt dann (2 Münzen)

Zeiterfassung:

Datum	Startzeit	Endzeit	Kommentar	Stunden
1.2.2024	9:00	14:00	Card, Package class implemented, structure for program	5
2. 2.2024	10:00	16:00	http Server implemented, server vertehen,	6

3. 2.2024	10:00	15:00	http Server Endpoints erstellt, User Managerklasse, Package und Card klasse expanded	5
4. 2.2024	11:00	15:00		4
5. 2.2024	9:00	13:00		4
6. 2.2024	8:00	15:00	Card, Battle, TradingDeal Adding and refactoring	7
7. 2.2024	10:00	15:30		5.5
8. 2.2024	11:00	14:00	Postgres DB created, linked	3
9. 2.2024	9:00	13:00	Battle Error correction, Unittest Implemented, Framework added	4
10. 2.2024	10:00	17:00		7
11. 2.2024	10:00	16:00		6
12. 2.2024	9:00	16:00	DB Connection for classes, SQL statements, Unittest 12/20 OK	7
13. 2.2024	10:00	13:00	Additional SQL Statements, 20/20 Unittest OK	3
14. 2.2024	9:00	14:00	UserEndpoint Created (Register OK), Understood how endpoints work on server	5
15. 2.2024	10:00	16:00	Session, Package, Deck, Transaction endpoint created, UserCardManager created,	6
16. 2.2024	10:00	15:00	Battle, Scoreboard, Stats Userendpoint implemented + helper class, Battlelogic DB connection	5
23. 2.2024	12:00	15:00	Curl Script: CreateUser, Login, Create Pack, Get Pack, edit user, display stats, display scores, Funktioniert. Battle: Startet noch logic Fehler	3
24. 2.2024	9:00	18:00		9
25. 2.2024	9:00	15:00	Curl Battle funktionert, Protocoll geschrieben, Layout gezeichnet, letzte Tests gemacht.	6

Gesamtdauer: 100.5 Stunden

Nicht geschafft: Trading deal, wurde zwar implementiert konnte aber nicht mehr an Endpoint angebunden werden.

GIT: https://github.com/Nickel0620/MonsterTradingCardGame