Describe design

Das DAO-Pattern ist ein bewährtes design pattern, das in der Softwareentwicklung häufig eingesetzt wird. Es bietet eine Möglichkeit, um die Anwendungslogik von der Datenbank zu trennen und vereinfacht dadurch die Handhabung von Daten und Datenbanken in einem Projekt.

Indem das DAO-Pattern angewendet wird, wird eine Abstraktionsschicht zwischen dem Anwendungscode und der Datenbank erstellt. Das bedeutet, dass die business logic der Anwendung von der Datenbank getrennt ist und somit Änderungen an der Datenbank ohne großen Einfluss auf den Anwendungscode durchgeführt werden können. Dies ermöglicht eine bessere Wartbarkeit und Flexibilität der Anwendung, was besonders bei komplexen Projekten von Vorteil ist.

Das DAO-Pattern bietet auch eine vereinfachte Schnittstelle für die Interaktion der Anwendung mit der Datenbank. Dadurch können Entwickler einfacher auf die Daten zugreifen und sie manipulieren, ohne sich um die spezifischen Details der Datenbank kümmern zu müssen. Das bedeutet, dass Entwickler sich auf die Implementierung der Geschäftslogik konzentrieren können, anstatt Zeit damit zu verschwenden, sich mit den Details der Datenbank auseinandersetzen zu müssen.

Zusätzlich verbessert das DAO-Pattern die Testbarkeit der Anwendung. Durch die Trennung der Datenbank von der Geschäftslogik wird es einfacher, automatisierte Tests durchzuführen und Fehler in der Anwendung zu finden. Es vereinfacht auch die Implementierung von Mock-Objekten, die für Integrationstests verwendet werden können.

Describe lessons learned

Dont call anything Package -> gitignore

Describe unit testing decisions

Die Tests verwenden Mock-Objekte, um den getesteten Code zu isolieren und sicherzustellen, dass die Tests wiederholbar und deterministisch sind. Die Tests decken sowohl Erfolgs- als auch Fehlerfälle ab und überprüfen erwartete Antworten und Fehlerbehandlung. Die Tests nutzen auch das NUnit-Testframework und seine integrierten Assertions, um das Schreiben und Lesen der Tests zu vereinfachen. Insgesamt zielen die Tests darauf ab, die Korrektheit der CardsManager-, GameManager-, UserManager-,TradingManager-,PackageManager und verwandten Klassen zu überprüfen.

Describes unique fearture

-

Tracked time

22.12.2022

* 1,5h
* Created Git repo & project

04.01.2023

* 2h
* Changes default project into given Message Server

24.01.2023

* 5h
* Deleted classes which are not needed & started setup Database &started with Database Daos

25.01.2023

* 4h
* Added functionallity to few commands for route & worked on Database

26.01.2023/27.01.2023

* 5,5h
* Finished basic functionallity for few commands in route & found critical issue in .gitignore

30.01.2023

* 2h
* Finished basic functionallity for few commands in route

15.02.2023

* 4h
* Added missing responses for all commands

23.02.2023

* 1h
* Fixed a bug where CardType parse didnt work

24.02.2023

* 4h
* Changed default result for the request to json & started battle endpoint

02.03.2023

* 3h
* Battle logic monster vs monster done, spell vs spell started

03.03.2023

* 1h
* Add bidirectional damage effekt for spell attacks

04.03.2023

* 2,5h
* Add Card-/Game-/UserTests and also half working package tests

05.03.2023

* 4,5h
* Add multiple tests covering allmost every command & Added protocol

link to GIT: https://github.com/Erdem-Sevilmis/MonsterTradingCardsGame