**Design:** The project is developed in C# with Postgres as the chosen data storage solution. Docker is used to run a Postgres image. The server functions as a RESTful service, managing incoming requests with threads. Tools:

* DBeaver for database management.
* Postman for API testing.
* Docker for containerization.
* Visual Studio as the IDE.
* GIT for version control.

Project Architecture:

1. Data Access Object (DAO)
2. Services
3. Models
4. Endpoints
5. Log Output (Unique Feature)

Endpoint classes handle invoked routes, parsing JSON bodies, and managing responses. These endpoints use services that interact with DAOs.

DAO: DAOs encapsulate logic for CRUD operations on the data store, providing a consistent API for the rest of the application.

Services: Service classes encapsulate complex business logic and rules, offering a simplified and consistent API for the application.

The battle logic involves creating a player in a queue and waiting for the next player. The implementation of editing user data and the profile page did not reach completion. Additionally, stats were implemented, but the scoreboard was not. Specialities were integrated but are not considered in battles. The implementation of Elo did not materialize.

**Unique Features:** The project's unique feature is a battle log that records each battle and writes it to Log Output. This allows showcasing and reviewing battles.

**Unit Tests:** The unit testing strategy for the project involves the creation of 21 unit tests, with a significant portion utilizing Mocks. Mocks play a crucial role in testing by mimicking the behavior of external components, such as databases, allowing for controlled and repeatable testing scenarios. These tests are distributed across three classes, namely **BattleTest**, **CardModelTest**, and **Tests**, each focusing on specific aspects of the MTCG application. The **BattleTest** class ensures the proper functioning of battle-related features, the **CardModelTest** class validates card type modeling, and the **Tests** class comprehensively checks various services and functionalities in the MTCG application.

**Lesson Learned:** Proper planning and structural consideration save time during development. Understanding how to perform unit tests and mocking is crucial, as well as acquiring proficiency in working with databases and RESTful services. Additionally, mastering bug handling and exploring alternative methods to mitigate these issues is essential for smoother development workflows.

**Spent Time:** Approximately 80 hours. **GIT LINK:** [HBK42/monstercardgame (github.com)](https://github.com/HBK42/monstercardgame)