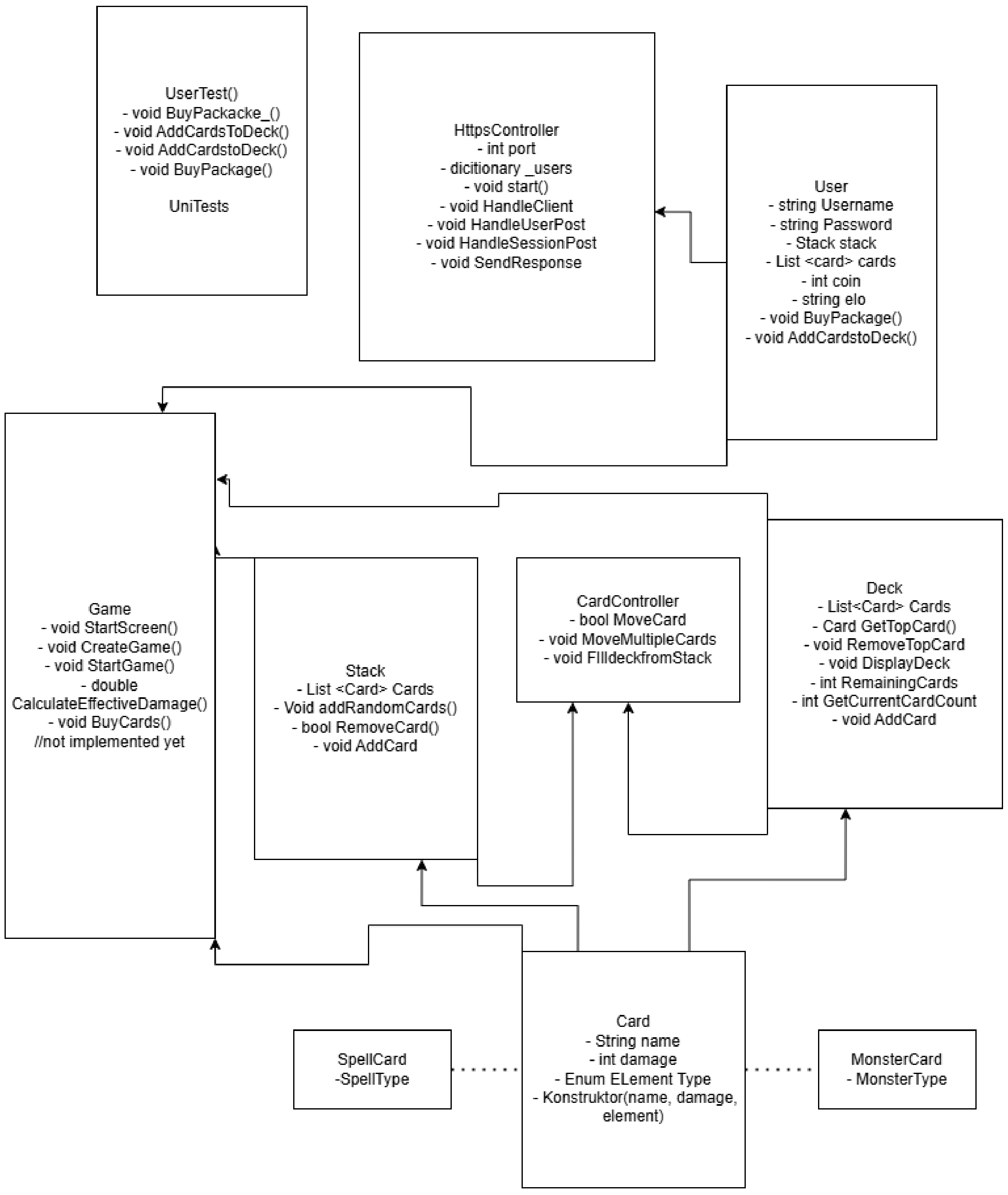

Short Description of the Programm

- During the Program run, a user is being Created, or you can login with new credentials that are created in the userCredentials.txt, so it getting stored locally and can be accessed manually in the root directory.
- While starting the Program, 2 Users are being created, that initializes the Gamelogic as well as starting the Game in the Backround. Both users Stack are getting filled up to 30 cards, which can be changed manually. Then during the Game, the deck is being filled with the cards from the stack. That is handled in the Cardcontroller and user class. After Then the first Cards from the deck are being compared and the more Dmg one wins, and the winner attains the card

from the 2nd user. Ther there might be a Infinite game loop, so therefore its limited to 100 rounds. Also each win is being counted with. So there is also a score Board in it.

AddRandomCards

- The AddRandomCards logic is in the Stack, where Cards are being Created with 50/50 Spell or MonsterCard. Each with a hopefully different name, added to their Prefix. Meaning Monster or SpellCard + number between 1-50. Also, the type and Element is added.
- The affinity Logig is being treated in the GameClass in the CalculateEffectiveDamage() method.

HttpsController

- **Functionality Overview**

The server supports two key actions:

- **User Registration (POST /users)**

Purpose: This allows you to create a new user by providing a username and a password.

How it works: When a POST request is made to /users with the appropriate data (username and password), the server checks if the user already exists. If not, it creates the new user and responds with a success message. If the user already exists or the data is invalid, it returns an error.

- **User Login (POST /sessions)**

Purpose: This allows users to log in by providing their username and password.

How it works: When a POST request is made to /sessions with the correct username and password, the server verifies the credentials. If they are correct, it returns a session token, which can be used for further authenticated requests. If the credentials are wrong, it responds with an error message.

- Testing it

So basically, I tested it with postman, because I'm familiar with it already. so how does it work? Basically, typing in the localhost(port) /endpoint, and then adding the body in the contentType to application/json data you can send in Requests. I used this to test the functionality.