Dokumentation Lernkarte Digitalisieren

Walser Sven

Universität St. Gallen

05.09.2022

2022

Inhaltsverzeichnis

[1 Zusammenfassung 2](#_Toc113284668)

[2 Planung 3](#_Toc113284669)

[2.1 Aufgabenstellung 3](#_Toc113284670)

[2.2 Anforderungen 3](#_Toc113284671)

[2.3 Klassendiagramm 5](#_Toc113284672)

[2.4 MockUp 6](#_Toc113284673)

[2.4.1 MainForm 6](#_Toc113284674)

[2.4.2 EditCard 6](#_Toc113284675)

[2.5 Tools 7](#_Toc113284676)

[3 Implementierung 7](#_Toc113284677)

[3.1 Card 7](#_Toc113284678)

[3.2 CardBox 7](#_Toc113284679)

[3.3 Access Data 9](#_Toc113284680)

[3.4 MainForm 10](#_Toc113284681)

[3.5 EditCard 10](#_Toc113284682)

[3.6 CardList 10](#_Toc113284683)

[4 Reflexion 11](#_Toc113284684)

# Zusammenfassung

Das Produkt dieses Projektes ist eine Applikation zum Lernen von Wörtern in Englisch zu Deutsch und Deutsch zu Englisch. Gelernt wird, indem ein zufälliges Wort ausgewählt wird und der Benutzer die korrekte Übersetzung aus einer Liste von Wörtern auswählt. Ist das ausgewählte Wort korrekt, so verschiebt es sich in einen höheren Slot, von denen es insgesamt 3 gibt. Ist die Auswahl falsch, so wird die Karte in einen niedrigeren Slot verschoben. Es beinhaltet einen vorhandenen Wortschatz von 51 Wörtern, welche in basis und erweitertem Wissen aufgeteilt sind. Dieser Wortschatz wird in einer Airtable Datenbank gespeichert, auf welche per REST zugegriffen wird. Dieser Wortschatz kann beliebig bearbeitet und erweitert werden. Bei der Applikation handelt es sich um eine Windows Forms Applikation, welche in C# geschrieben wurde. Das Projekt wurde von einem Praktikanten mit einer Dauer von 88 Arbeitsstunden durchgeführt.

# Planung

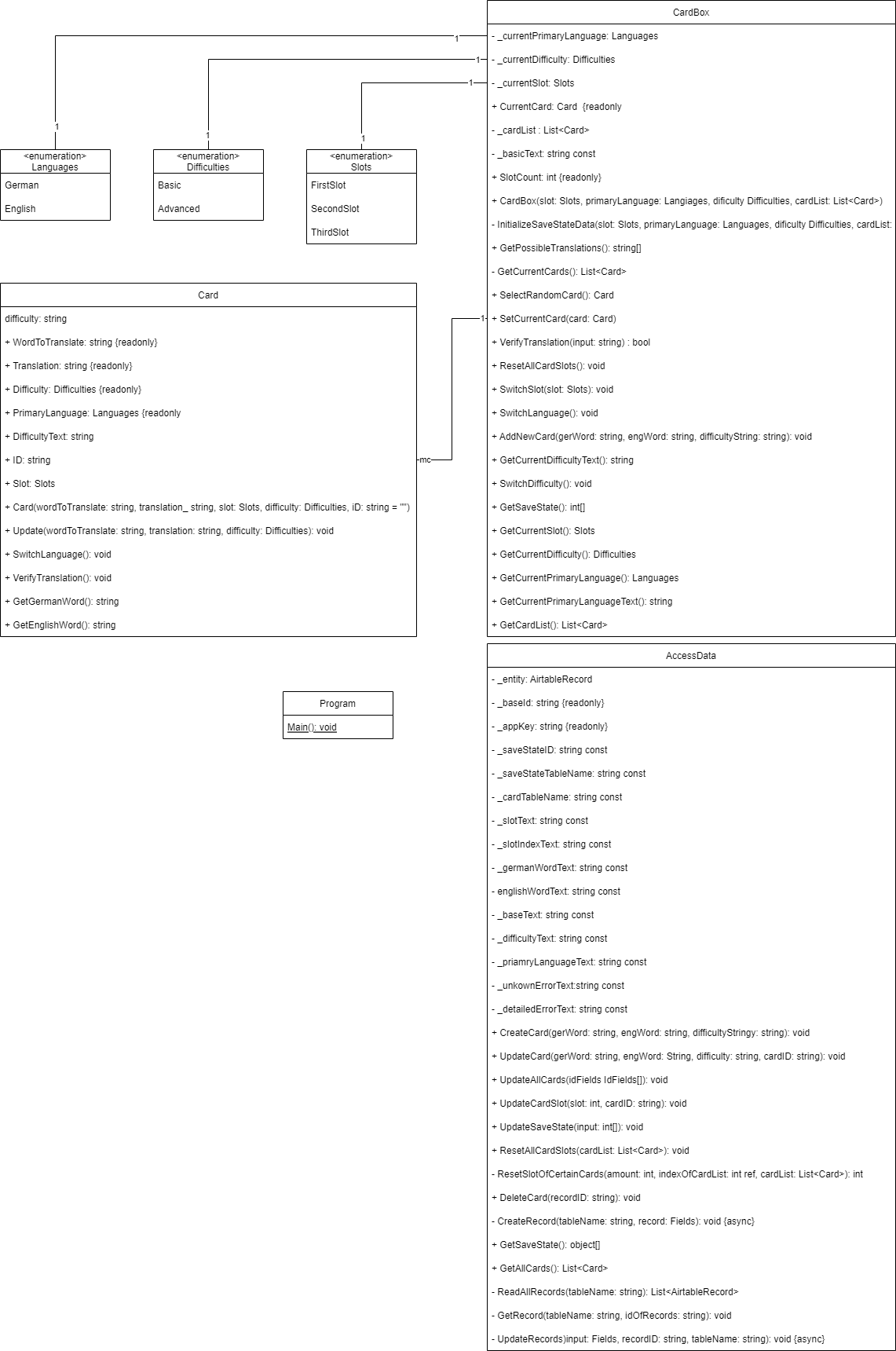
## Aufgabenstellung

Es soll eine Anwendung geschrieben werden, welche die physischen Lernkarten ersetzen soll. Dabei soll es zwei Kategorien unterschieden werden: basis- und erweitertes Wissen. Die Anwendung zeigt aufgrund der Einstellung (basis oder erweitertes Wissen) eine Karte. Die Anwendung soll einstellbar sein, was abgefragt werden soll: Deutsch oder Englisch. Danach wählt man aus einer Liste die Übersetzung. Wenn man es richtig erraten hat, dann verschwindet es aus der Liste. Das Lernen soll in Slots gelernt werden: der erste Slot 1-mal richtig, der zweite Slot 2-mal richtig, der dritte Slot 3-mal richtig. Beispielhaft kann man es so beschreiben: jedes Mal richtig geraten, verschiebt es um einen Slot rechts und wenn man falsch war, dann verschiebt es um einen Slot links. Die Reihenfolge der Auswahl soll auf zwei Arten erfolgen: Nacheinander und zufällig.  
Das Erfassen neuer Wörter soll auch ermöglicht werden. Die Daten werden in einem File gespeichert.

## Anforderungen

|  |  |
| --- | --- |
| **Anforderung** | **Gewichtung** |
| Ein zu übersetzendes Wort automatisch auswählen | Muss |
| Eine Liste aus auswählbaren Begriffen anzeigen und den Richtigen auswählen können | Muss |
| Begriff verschwindet bei korrekter Auswahl aus der Liste | Muss |
| Bei Korrekter Auswahl der Übersetzung wird ein neues zu übersetzendes Wort wird angezeigt | Muss |
| Begriffe in eine Datenbank speichern | Muss |
| Bei korrekter Auswahl der Übersetzung des Begriffes, wird die Karte in einen höheren Slot verschoben | Muss |
| Bei falscher Auswahl der Übersetzung eines Begriffes, wird die Karte in den nächsttieferen Slot verschoben | Muss |
| Begriffe in den Slots erfolgen nacheinander und sollen in zufälliger Reihenfolge erscheinen | Muss |
| Zwischen Deutsch und Englisch Übersetzungen wechseln können | Soll |
| Zwischen Basis und erweitertem Wissen wechseln können | Soll |
| Es sollen keine Fehleingaben möglich sein oder ansonsten abgefangen werden | Soll |
| Neue Begriffe hinzufügen können | Soll |
| Unit-Tests Durchführen | Soll |
| Lernfortschritt speichern | Kann |
| Lernfortschritt zurücksetzen | Kann |
| Liste von allen Karten und deren Begriffen | Kann |
| Karten wieder löschen | Kann |

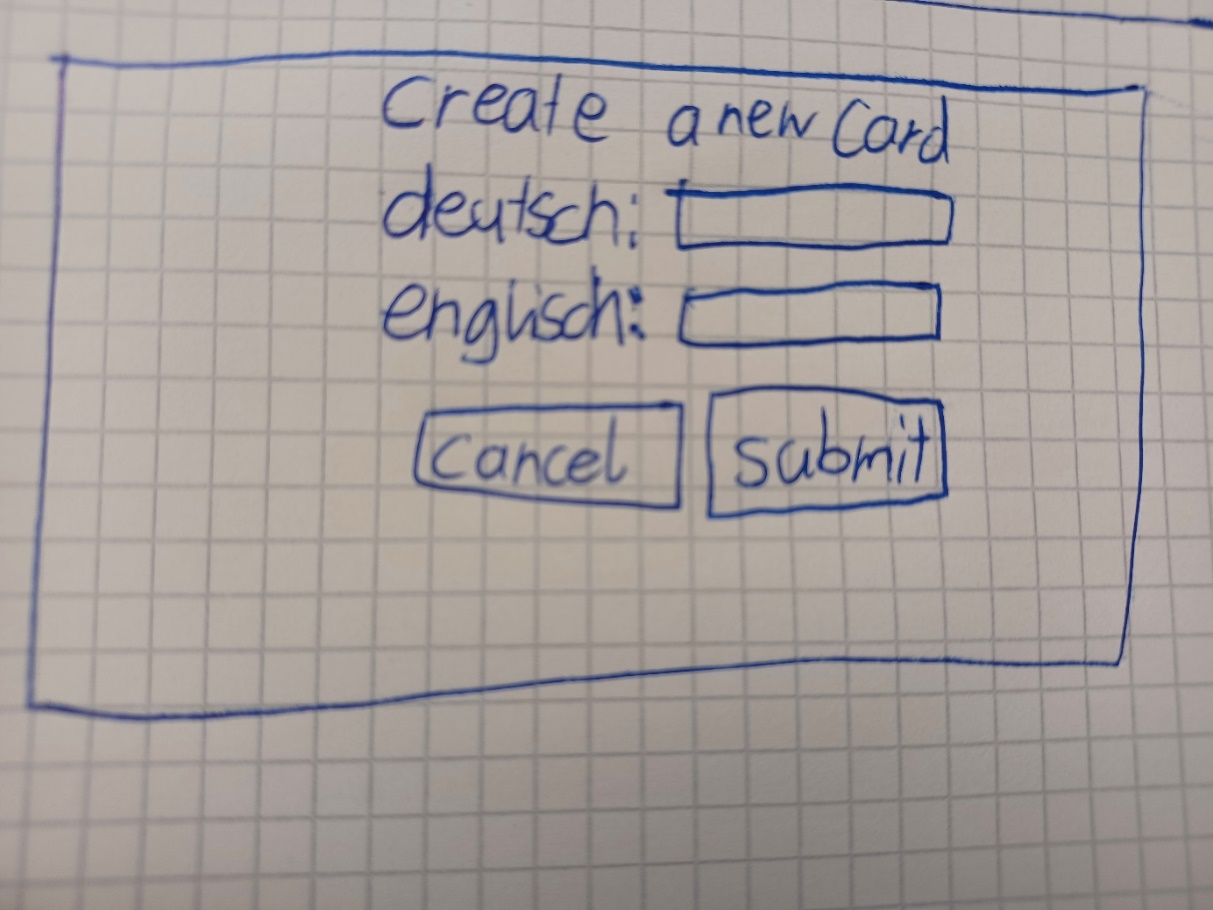
## Klassendiagramm



## MockUp

### MainForm

### EditCard



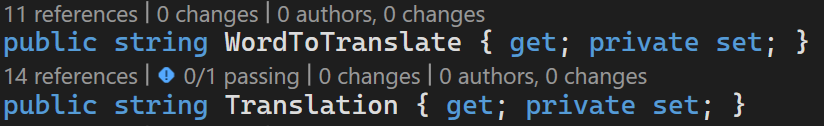
## Tools

* Visual Studio 2022 – Implementierung des Projektes
* Azure DevOps – Notierung der Aufgaben
* Microsoft Excel – Notierung der Zeit der Aufgaben
* Microsoft Word – Dokumentation des Projektes
* GitHub – Speicherung der Daten
* Git – Speicherung der Daten

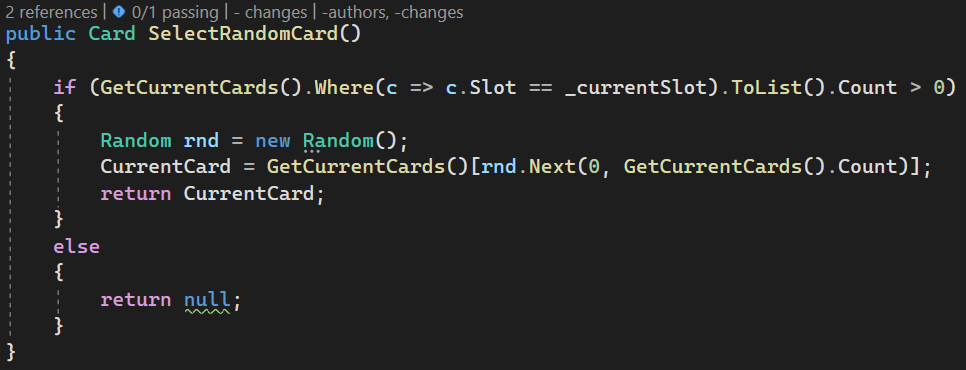
# Implementierung

## Card

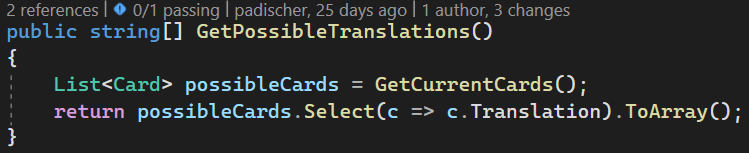
Das Programm basiert auf sogenannten Cards welche Lernkarten darstellen sollen. Deren wichtigsten Eigenschaften sind WordToTranslate und Translation, da sie das deutsche sowie englische Wort darstellen.



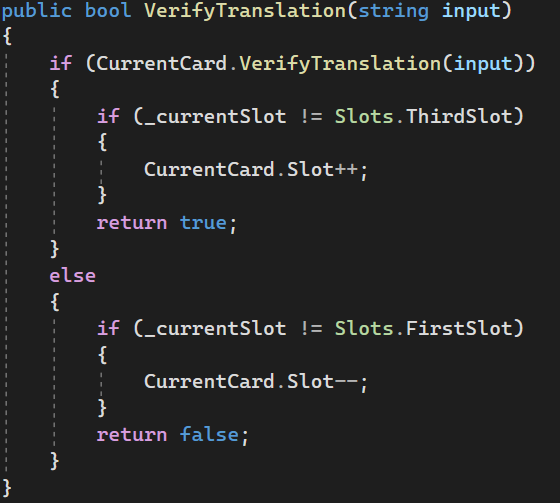
## CardBox

CardBox stellt das Herz der Logik dar. In dieser Klasse werden die Cards gespeichert und verwaltet. SelectRandomCard(), GetPossibleTranslations() und VerifyTranslation() sind die wichtigsten Funktionen. Mit SelectRandomCard() wird zufällig eine Karte aus dem aktuellen Slot ausgewählt und sie wird als aktuelle Karte gesetzt, was bedeutet, dass sie auf dem UI als zu übersetzende Karte angezeigt wird. 

Mit GetPossibleTranslations() werden alle Übersetzungen, welche sich im aktuellen Slot befinden sowie der momentan ausgewählten Sprache und Schwierigkeit entsprechen zurückgegeben.

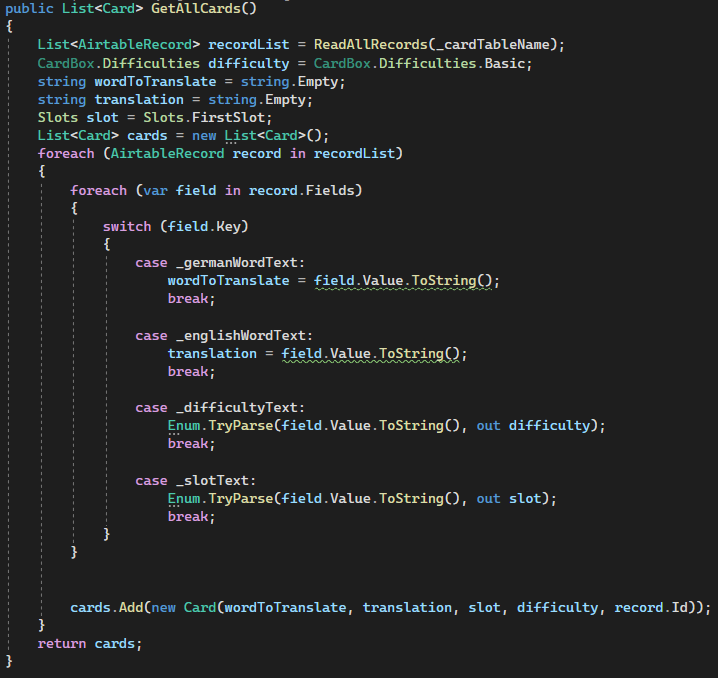


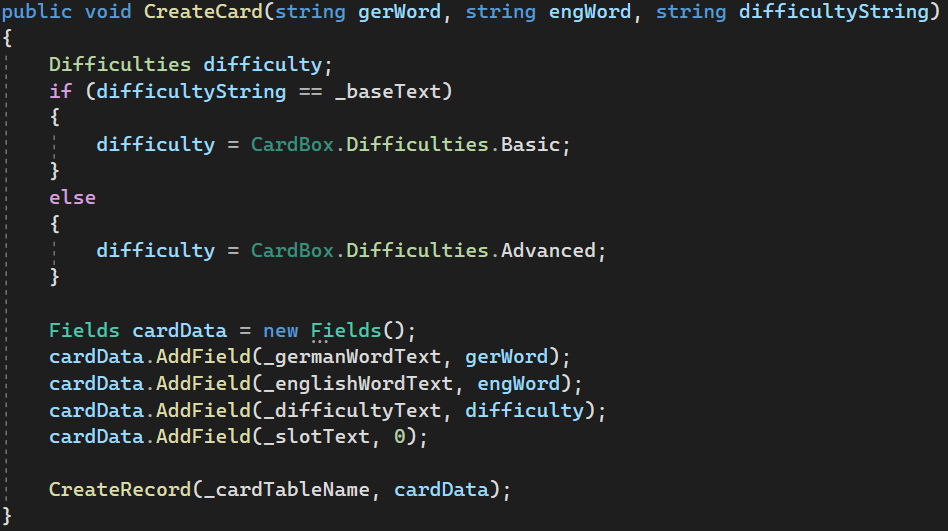
In der Funktion VerifyTranslation() wird ausserdem überprüft, ob die ausgewählte Übersetzung der tatsächlichen Übersetzung der aktuellen Karte entspricht.



Hier existiert ausserdem ein Slotsystem, welches aus 3 Slots besteht. Wird eine Karte korrekt gelernt, wird sie in einen höheren Slot verschoben. Sollte das falsche Ergebnis gewählt worden sein, verschiebt sich die Karte in einen tieferen Slot.

## Access Data

Die AccesssData Klasse regelt die Kommunikation mit der Datenbank API. Hier werden vom UI Anfragen an gewisse Daten geschickt und die Klasse leitet sie für die API verständlich weiter. Die wichtigsten Methoden in dieser Klasse sind GetAllCards() und CreateCard(). GetAllCards() holt alle Cards aus der Datenbank und gibt sie zurück. 

CreateCard() Create Card erstellt eine neue Karte in der Datenbank und übergibt das deutsche Wort, das englische Wort und die Schwierigkeit als Parameter. Der Slot wird automatisch auf 1 gesetzt. 

## MainForm

Das MainForm ist die Startseite, auf der die Wörter gelernt werden. Hier befinden sich ein zu übersetzendes Wort, eine Liste mit möglichen Übersetzungen, eine Drop Down Liste um zwischen den Slots zu wechseln, Buttons und zwischen der Sprache und der Schwierigkeit zu wechseln, das Programm zu beenden, um alle Cards in den ersten Slot zu verschieben, um auf das EditCard Form zu gelangen, um das CardList Form anzuzeigen und um das ausgewählte Wort aus der Liste zu bestätigen.

## EditCard

Das EditCard Form hat zwei verschiedene Textfelder für das englische und deutsche Wort einer Card sowie eine DropDownList für die Schwierigkeit. Zum Schluss gibt es noch 2 Buttons, um die Eingabe zu bestätigen und mit den Daten eine neue Karte zu erstellen oder das Verfahren abzubrechen und das Form zu schliessen. Wenn das Form über CardList und nicht das MainForm geöffnet wird, erfüllt es eine andere Funktion. So werden die beiden Textfelder und die DropDownListe automatisch mit den Daten der ausgewählten Karte ausgefüllt und ein Klick auf den Bestätigen-Button speichert Änderungen, welche vorgenommen wurden.

## CardList

Das CardList Form beinhaltet eine Liste von allen Cards, deren Begriff auf Deutsch und Englisch sowie deren Schwierigkeit. Es gibt einen Button zum Löschen einer ausgewählten Card, zum Aufruft von EditCard um die ausgewählte Card zu bearbeiten und einen Button um das Fenster zu schliessen und zurück auf das MainForm zu gelangen.

# Reflexion

Ich habe die Erkenntnis gemacht, dass ich mehr Zeit für Projekte einplanen muss und dies aus mehreren Gründen. Es können immer unvorhergesehene Probleme vorkommen, welche die Dauer des Auftrags erheblich beeinflussen können. Ausserdem ist es mehrmals vorgekommen, dass sich Anforderungen verändert haben oder neue Anforderungen hinzugekommen sind. Zum Schluss sollte für die Aufgaben mehr Zeit eingeplant werden, damit man sich mehr Zeit beim für das schöne Entwickeln des Codes lassen kann, um den Refactoring Aufwand zu verringern. Dies hat den Vorteil, dass man später nicht teilweise mehrere Klassen umschreiben muss.

Ich bin mir über die Wichtigkeit des Clean Code bewusst geworden. An einigen Zeitpunkten der Entwicklung habe ich Methodennamen öfters verwendet und den Überblick grösstenteils verloren. Ausserdem hatte ich zu einem früheren Zeitpunkt den Zugriff auf die Datenbank teils in der CardBox Klasse implementiert, was es mir nicht ermöglichte dies gut zu testen. Somit hätte ich bei jedem Test eine Bearbeitung an der Datenbank vorgenommen. Dieses fehlende Auslagern der Business Logik ist im Programm noch häufiger vorgekommen. Nun habe ich auch noch den Fehler gemacht, oftmals keine Aussagekräftigen Namen zu geben. So gab es Methoden in der gleichen Klasse, welche so aussahen: TestGetCurrentDifficultyText, TestGetCurrentDifficultyText1.

Zuletzt habe ich mich noch mit einigen Tools vertraut gemacht, welche ich vorher nur selten verwendet habe. Diese sind vor allem Git und Azure DevOps.