FIAE20J – LF12a & Deutsch

Fachinformatiker für Anwendungsentwicklung

Dokumentation zum Projekt



**Avalanche**

**Programm zur Ansicht und Verwaltung von Snowboardern**

Abgabetermin: 13.11.2022

# Inhaltsverzeichnis

[Inhaltsverzeichnis I](#_Toc113366748)

[Abkürzungsverzeichnis II](#_Toc113366749)

[1 Einleitung 1](#_Toc113366750)

[1.1 Projektbeschreibung 1](#_Toc113366751)

[1.2 Projektziel 1](#_Toc113366752)

[2 Projektplanung 1](#_Toc113366753)

[2.1 Ressourcenplanung 1](#_Toc113366754)

[2.2 Entwicklungsprozess 1](#_Toc113366755)

[3 Entwurfsphase 1](#_Toc113366756)

[3.1 Git 1](#_Toc113366757)

[3.2 Entwurfsmuster 1](#_Toc113366758)

[3.3 Entwurf der Benutzeroberfläche 1](#_Toc113366759)

[4 Implementierungsphase 2](#_Toc113366760)

[4.1 Git 2](#_Toc113366761)

[4.2 Implementierung von MVC 2](#_Toc113366762)

[4.3 Implementierung der Benutzeroberfläche 2](#_Toc113366763)

[4.4 Implementierung der Datenbank 2](#_Toc113366764)

[4.5 Abweichungen gegenüber dem erwarteten Ergebnis 2](#_Toc113366765)

[5 Dokumentation 2](#_Toc113366766)

[6 Fazit 2](#_Toc113366767)

[6.1 Soll-/Ist-Vergleich 2](#_Toc113366768)

[A Anhang i](#_Toc113366769)

[A.1 Abbildung ER-Diagramm Datenbank i](#_Toc113366770)

# Abkürzungsverzeichnis

ISC *International Snowboarding Community*

MVC *Model-View-Controller*

SoC Separation of Concerns

# Einleitung

Die folgende Projektdokumentation beschreibt den Ablauf unseres Snowboarder Projekts, welches wir für die Notenvergabe in Deutsch und Lernfeld 12 anfertigen.

## Projektbeschreibung

Um die Wettkämpfe der Snowboarder zu Verwalten hat die International Snowboarding Community ein Programm in Auftrag gegeben. Die genaue Aufgabenbeschreibung ist im Anhang A.2 hinterlegt.

## Projektziel

Ziel des Projekts ist es ein Programm zu entwickeln, welches die Wettkämpfer, deren Sponsoren sowie andere relevante Informationen speichern und ausgeben kann. Geplant ist dafür die Verwendung einer Datenbank zum speichern der Informationen, sowie eine Möglichkeit diese Benutzerfreundlich zu bedienen.

# Projektplanung

## Ressourcenplanung

Für die Umsetzung des Projektes stand uns die Zeit bis zum 13 November zur Verfügung. Das Verfügbare Personal besteht dabei aus Simon, Vladimir und Can.  
Aufgrund der nahenden Abschlussprüfungen wurde im Team vereinbart den 16 September als neue Frist zu nutzen um die Abgabe frühzeitig von unserer To-Do Liste zu streichen.  
Als Entwicklungstools nutzen wir Visual Studio und SQLite.

## Entwicklungsprozess

Um die Entwicklung und das Projekt möglichst zügig voranzutreiben, wurde eine Teilung der Arbeit vereinbart.  
Die Teilung folgte in die folgenden Bereiche: Dokumentation, Datenbank und Benutzeroberfläche. Diese Verantwortlichkeiten haben wir bei uns im Team verteilt. Keiner der Teile kann dabei Selbstständig funktionieren, sodass ein ständiger Austausch im Team und der aktuellen Arbeit gewährleistet ist.

# Entwurfsphase

## Git

Als Quellcodeverwaltungssystem wird Git verwendet. Das Projekt soll in mehreren Git-Branches entwickelt werden um einen parallelen Fortschritt zu ermöglichen. Das Ganze wird dabei in der Cloud gespeichert, mit GitHub als freizugänglichen Dienstleister.

## Entwurfsmuster

Zu Beginn der Aufgabe haben wir uns zudem auf ein Entwurfsmuster festgelegt, welches wir für die Architektur des Projekts nutzen wollen. Hierbei ist die Wahl auf das Model-View-Controller (MVC) Muster gefallen. Der Vorteil des MVC-Musters ist Separation of Concerns (SoC). Hierbei wird das Projekt in Komponenten mit klar definierten Aufgaben geteilt. Durch die Teilung können Änderungen und Erweiterung an einer Komponente durchgeführt werden ohne die Funktion der anderen Komponenten zu beeinflussen. Weiterhin wird das Projekt übersichtlicher strukturiert und einfacher zu Debuggen.

Zusätzlich zu dem MVC-Muster haben wir uns für eine Implementierung des Repository-Musters in Kombination mit dem Unit of Work-Muster entschieden. Das Repository-Pattern bietet hierbei eine weitere Abstraktionsebene zwischen dem Controller und der Datenbank. Der Vorteil ist hier erneut SoC. Durch die zusätzliche Abstraktionsebene mit generischen Methoden für den Datenzugriff sind Änderungen der Persistenzschicht, also der Datenbank oder einer ähnlichen Datenquellen wie z.B. eine CSV-Datei oder eine NoSQL-Datenbank, möglich ohne den Rest des Projekts zu beeinflussen. Weiterhin kann die Anzahl an Code-Duplikaten durch Datenbankaufrufe stark reduziert werden.

Das Unit of Work-Muster arbeitet hier mit dem Repository-Muster zusammen und erlaubt es uns mehrere Datenbank-Operationen zu einer einzelnen Transaktion zusammenzufassen. Hierdurch werden Datenfehler durch nur teilweise durchgeführte Commits vermieden, da entweder alle Operationen einer Transaktion erfolgreich sein müssen oder ein Rollback durchgeführt wird. Gleichzeitig übernimmt das Unit of Work den Zugriff auf die Datenbank und sorgt erneut für eine weitere SoC.

## Entwurf des Datenmodells

Das Datenmodell für die Anwendung wurde anhand den Anforderungen in der Aufgabenbeschreibung (siehe Anhang A.2) erstellt. Hierfür wurde zuerst ein ER-Modell erstellt, welches anschließend in die 3. Normalform überführt wurde. Das finale ER-Modell ist im Anhang A.1 hinterlegt.

## Entwurf der Benutzeroberfläche

Die Benutzeroberfläche wird als ASP.net Anwendung umgesetzt. Dies erlaubt den Zugriff auf die Datenbank von verschiedensten Geräten. Da die ISC eine Internationale Vereinigung ist, erlaubt unser Ansatz für eine Einfache Erreichbarkeit unserer Services Weltweit.  
Optisch wird die Anwendung in einem klassischen Layout mit einer Navigationsleiste am oberen Rand des Bildschirms aufgebaut.

Logisch wird die Anwendung in drei Teile aufgeteilt. Snowboarder, Wettkämpfe und Misc (Sonstiges). Wir haben uns für diese Aufteilung entschieden, da die Tabellen Snowboarder und Wettkampf die zentralen Tabellen des gegebenen Datenmodells und damit der Anwendung sind. Alle weiteren Tabellen des Datenmodells speisen diese zentralen Tabellen und wurden somit unter der Kategorie Misc zusammengefasst.

Das Hinzufügen und Ändern von Daten soll auf den jeweiligen Seiten über Formulare erfolgen. Die anschließende Darstellung der jeweiligen Daten geschieht über eine tabellarische Ansicht in den jeweiligen Kategorien.

## Entwurf der C#-Klassen

Zum Entwurf der notwendigen C#-Klassen wurde ein Klassendiagramm (siehe Anhang A.?) erstellt. Hierbei wurde mit den Klassen für die Modelle, welche von den Views verwendet werden, begonnen.

# Implementierungsphase

## Git

Zu Beginn der Implementierungsphase wurde für ein ASP .NET MVC-Projekt in Visual Studio erstellt, und dann in die Cloud von GitHub gepusht. Anschließend wurden die nötigen Rechte an alle Teammitglieder vergeben. Dem Basisprojekt folgten 3 Branches um die Fortschritte getrennt zu erarbeiten. Benötigte Änderung konnten so durch einen Merge der Branches oder einen Cherry-Pick einzelner Commits zwischen den Teammitgliedern ausgetauscht werden.

## Implementierung von MVC

Die grundlegenden Funktionalitäten des MVC-Musters wurden als Teil der Erstellung des ASP .NET MVC-Projekts in der Program.cs durch ASP .NET eigene Middleware implementiert.

## Implementierung der Benutzeroberfläche

~~Ein Screenshot der implementierten Benutzeroberfläche befindet sich im Anhang A.17: Screenshot der Benutzeroberfläche auf Seite xii.~~

## Implementierung der Datenbank

~~Für alle im Folgenden angelegten Klassen stehen Basis-Klassen aus dem hauseigenen Core-Framework zur Verfügung, durch die grundlegende Funktionalitäten bereitgestellt werden.~~

## Abweichungen gegenüber dem erwarteten Ergebnis

~~Aufgrund der parallelen Entwicklung an der Firmware und dem intensiven Austausch mit dem~~ [~~DDE~~](#_Abkürzungsverzeichnis) ~~ist es zu Verzögerungen in der Implementierungsphase gekommen. Die Implementierung der Tests wurde zum Teil auf einen späteren Zeitpunkt verschoben.~~

# Dokumentation

Um die Zeitlichen Anforderungen einzuhalten wurde die Dokumentation parallel zur Entwicklung geschrieben.

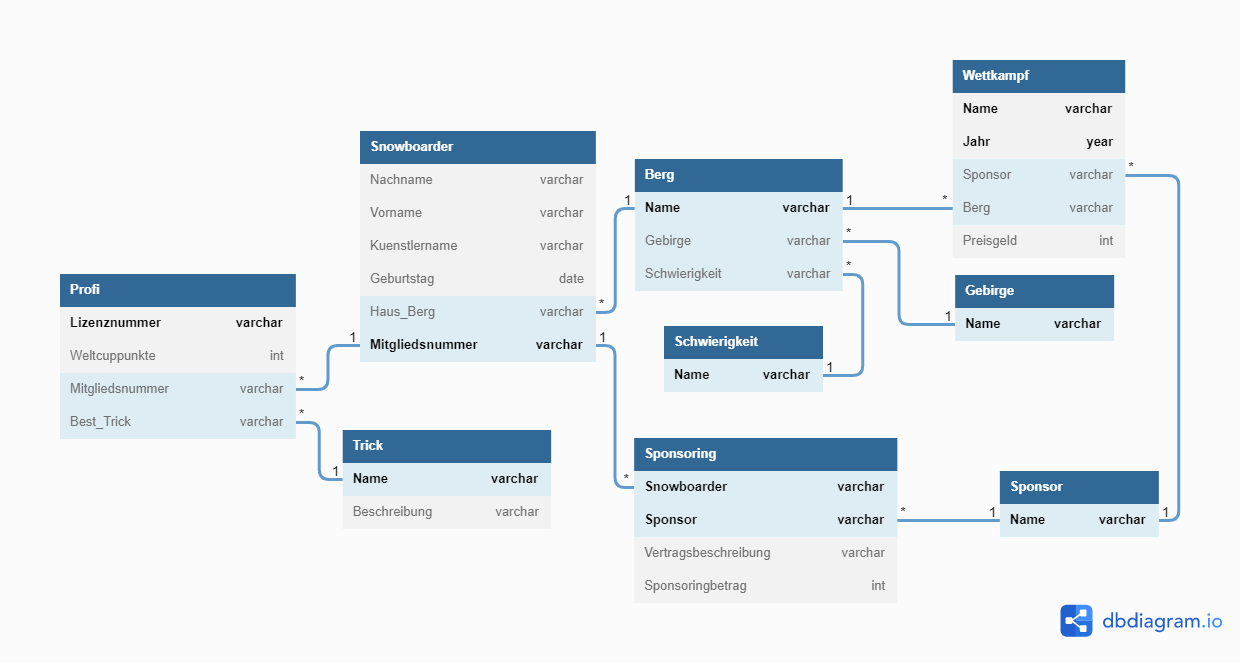
# Fazit

Zum Abschluss des Projekts zieht der Autor ein Fazit über das Gelernte und gibt einen Ausblick auf die Zukunft des Projekts.

## Soll-/Ist-Vergleich

~~Das~~ [~~SMCU5~~](#_Abkürzungsverzeichnis) ~~soll in unterschiedlichen Projekten mit unterschiedlichen Maschinen verwendet werden. Zum Zeitpunkt dieser Dokumentation werden Firmware und Maschinenkonfigurationen weiterhin erarbeitet. Die parallele Entwicklung des~~ [~~SMCU5~~](#_Abkürzungsverzeichnis) ~~und der stetige Austausch mit dem~~ [~~DDE~~](#_Abkürzungsverzeichnis) ~~haben die Implementierungsphase verlängert. Es wurde die Entscheidung getroffen, die Unit-Tests für den~~ [~~UI~~](#_Abkürzungsverzeichnis)~~-Service zu einem späteren Zeitpunkt zu implementieren.~~

1. Anhang
2. Abbildung ER-Diagramm Datenbank



1. Aufgabenbeschreibung

Du hast den Auftrag bekommen, für die International Snowboarding Community (ISC) ein Programm zur Verwaltung aller Wettkämpfe zu erstellen. Nach langwierigen Gesprächen und zahlreichen Après-Ski-Veranstaltungen ist es dir gelungen, die Daten zu identifizieren, die für die ISC wichtig sind:

Snowboarder haben einen Nachnamen, einen Vornamen, einen Künstlernamen und auch einen Geburtstag. Jedem ist eine eindeutige Mitgliedernummer zugeordnet.

Zusätzlich soll für jeden Snowboarder der "Haus-Berg" abrufbar sein. Dieser liegt in einem Gebirge und hat eine von der ISC vergebene Schwierigkeitsstufe.

Unter den Snowboardern gibt es Profis. Diese haben eine eigene Lizenznummer, Weltcup-Punkte und ihren "Best-Trick". Zudem haben sie mindestens einen Sponsor, von dem sie mit einem bestimmten Betrag finanziert werden.

Die Sponsoren, von denen lediglich der Name und ihre Sponsoringverträge bekannt sind, sind gleichzeitig auch die Veranstalter der Wettkämpfe. Dabei wird jeder Wettkampf von lediglich einem Sponsor an einem bestimmten Berg ausgetragen. Wettkämpfe werden mit dem Namen und dem Veranstaltungsjahr identifiziert. Bei jedem Wettkampf werden Preisgelder in unterschiedlicher Gesamthöhe ausgeschüttet.