FIAE20J – LF12a & Deutsch

Fachinformatiker für Anwendungsentwicklung

Dokumentation zum Projekt

Avalanche

Programm zur Ansicht und Verwaltung von Snowboardern

Abgabetermin: 13.11.2022

Inhaltsverzeichnis

Abkürzungsverzeichnis

ASP *Active Server Pages*
CSS *Cascading Style Sheets*
ISC *International Snowboarding Community*
O/RM *Object/Relational Mapping*
MVC *Model-View-Controller*
SoC *Separation of Concerns*

1 Einleitung

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

1.1 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.

1.2 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.

2 Projektplanung

2.1 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.
Als Entwicklungstools nutzen wir Visual Studio und SQLite.

2.2 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.

3 Entwurfsphase

3.1 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.

3.2 Zielplattform

Für die Umsetzung des Projekts haben wir uns für die Programmiersprache C# entschieden, da alle Entwickler des Teams gute Kenntnisse in dieser Programmiersprache haben. Als Framework zur Umsetzung von Webanwendungen bietet C# das Microsoft eigene Framework ASP.NET, welches wir zur Umsetzung nutzen werden. Als Zielframework haben wir .NET 6 gewählt, da es sich hierbei um die aktuellste Version mit langfristiger Unterstützung handelt.

Als Datenbank haben wir eine SQLite-Datenbank gewählt, da diese für die geforderten Anforderungen ausreichend ist und eine einfache Einrichtung mit geringem Overhead bietet. Für den Datenbankzugriff haben wir uns entschieden das Object/Relational Mapping (O/RM) Framework Entity Framework zu nutzen, da es sich hierbei um das de facto Standard O/RM-Framework für C# handelt und Teile des Teams hiermit bereits Erfahrungen hatten.

Zur optischen Aufwertung der Benutzeroberflächen haben wir uns entschieden das CSS-Framework Bootstrap zu nutzen, da dieses weitverbreitet ist und als Teil des Standard ASP.NET Projekts geliefert wird.

3.3 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.

Eine Darstellung des durch die Entwurfsmuster resultierenden Schichten-Modells ist in Anhang A.3 zu finden.

3.4 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.

3.5 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.

3.6 Entwurf der C#-Klassen

Zum Entwurf der notwendigen C#-Klassen wurde ein Klassendiagramm (siehe Anhang A.4) erstellt. Hierbei wurde mit den Klassen für die Modelle, welche von den Views verwendet werden, begonnen. Um diese Modelle von den späteren Datenbank-Modell-Klassen zu unterscheiden und zu zeigen, dass es sich um Modelle für die Views handelt, wurde sich für den Suffix ViewModel entschieden. Anschließend wurden drei Controller für die im vorherigen Abschnitt definierten Teile der Anwendung entworfen. Pro verfügbarer Seite benötigte der Controller eine dazugehörige Methode. Bei den Methoden der Controller musste darauf geachtet werden das für alle Seiten, die Daten an den Controller zurücksenden, zwei Versionen der gleichen Methode benötigen. Eine Version der Methode erhält Daten zur Darstellung vom Controller, die andere sendet eingegebene Daten an den Controller.

Nach dem Entwurf der Controller wurden die Klassen zur Implementierung des Repository-Musters und des Unit of Work-Musters entworfen. Beim Entwurf der Repository Klasse wurde außerdem ein Interface entworfen, um eine generalisierte Basis-Klasse des Repositories zu erstellen und es einfacher zu machen zukünftig weitere Repositories hinzuzufügen.

Die Datenbank-Modell-Klassen wurden durch den DB-Scaffold Befehl von Entity Framework erstellt und in das Klassendiagramm übertragen.

4 Implementierungsphase

4.1 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.

4.2 Projektstruktur

Die logische Struktur des Projekts wurde wie folgt aufgebaut. Im Projekt-Root befinden sich eine Reihe an Ordnern, die unterschiedliche Teile der Anwendung repräsentieren. Im Ordner Controller sind alle Klassen-Dateien der Controller zu finden. Der Data Ordner enthält alle Datenbank-Modell-Klassen sowie den DBContext, eine Entity Framework Klasse, über die auf die Datenbank zugegriffen wird. Der Ordner Doku enthält diese Projektdokumentation. Im Models-Ordner sind alle ViewModel-Klassen zu finden. Repositories enthält alle Repository-Klassen sowie die UnitOfWork-Klasse. SQL beinhaltet zwei SQL-Skripte zum Erstellen und Leeren der SQLite Datenbank. Zum Abschluss enthält Views alle .cshtml-Seiten, gruppiert nach den dazugehörigen Controllern.

4.3 Implementierung der Datenbank

Anhand des ER-Diagramms (siehe Anhang A.1) haben wir die SQL-Statements für die einzelnen Tabellen erstellt. Nachdem wir über das SQLite Kommandozeilentool eine leere Datenbank, die snowboarding.db, erstellt haben, wurden diese SQL-Statements in der Datenbank ausgeführt und alle benötigten Tabellen erstellt. Anschließend haben wir mittels Entity Framework die C# Datenbank-Modell-Klassen anhand der Datenbank generieren lassen.

4.4 Implementierung der C#-Klassen

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. Anschließend wurden die

notwendigen Controllerklassen erstellt und müssen für ihre Funktion von der ASP.NET-Klasse Controller erben.

Als nächstes wurden die Repositories erstellt. Hierbei wurden zunächst das Interface und die generische Basisklasse erstellt.

Nach den Repositories haben wir die UnitOfWork-Klasse erstellt, da diese die Repository-Klassen instanziiert. Die UnitOfWork-Klasse wurde anschließend jedem Controller als Objekt hinzugefügt.

Für die einzelnen Methoden der Controller-Klassen sind wir Seite für Seite vorgegangen und haben dabei abwechselnd die Methode und danach die Benutzeroberfläche implementiert. Während der Implementierung der Methoden ist aufgefallen das für einige Datenbank Entitäten spezialisierte Repository-Klassen notwendig sind. Diese wurden anschließend erstellt und erben vom Basis-Repository.

4.5 Implementierung der Benutzeroberfläche

Die Benutzeroberflächen wurden, wie im vorherigen Abschnitt beschrieben, im Wechsel mit den dazugehörigen Controller-Methoden implementiert. Hierbei blieben die Navigationsleiste im oberen Bereich und der Footer im unteren Bereich der Anwendung auf allen Seiten gleich. Diese beiden Elemente wurden als geteilte Elemente implementiert und mussten so nicht auf jeder Seite neu angelegt werden.

Die eigentlichen Elemente der Seite wurden im freien Bereich in der Mitte der Anwendung angezeigt. Eingabemasken für den Nutzer wurden hier in Form von Formularen mit einem Speichern Button umgesetzt. Die Dropdownmenüs einiger Formulare, wie z.B. dem Snowboarder hinzufügen Formular, werden hierbei mit relevanten Daten aus der Datenbank gefüllt, aus denen der Nutzer auswählen kann.

Zur Darstellungen der Daten haben wir uns für eine tabellarische Präsentation der Daten entschieden. Hierbei stehen bei jedem Datensatz in der Tabelle zusätzliche Optionen, wie die Bearbeitung der Daten oder eine Detailansicht mit zusätzlichen Daten, zur Verfügung.

4.6 Abweichungen gegenüber dem erwarteten Ergebnis

Für das Projekt wurde nur eine Eingabe und eine Präsentation der Daten gefordert. Um die Anwendung jedoch abzurunden haben wir uns für die zusätzliche Implementierung der restlichen CRUD-Funktionen, dem Bearbeiten vorhandener Daten und das Löschen von Daten entschieden.

5 Dokumentation

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

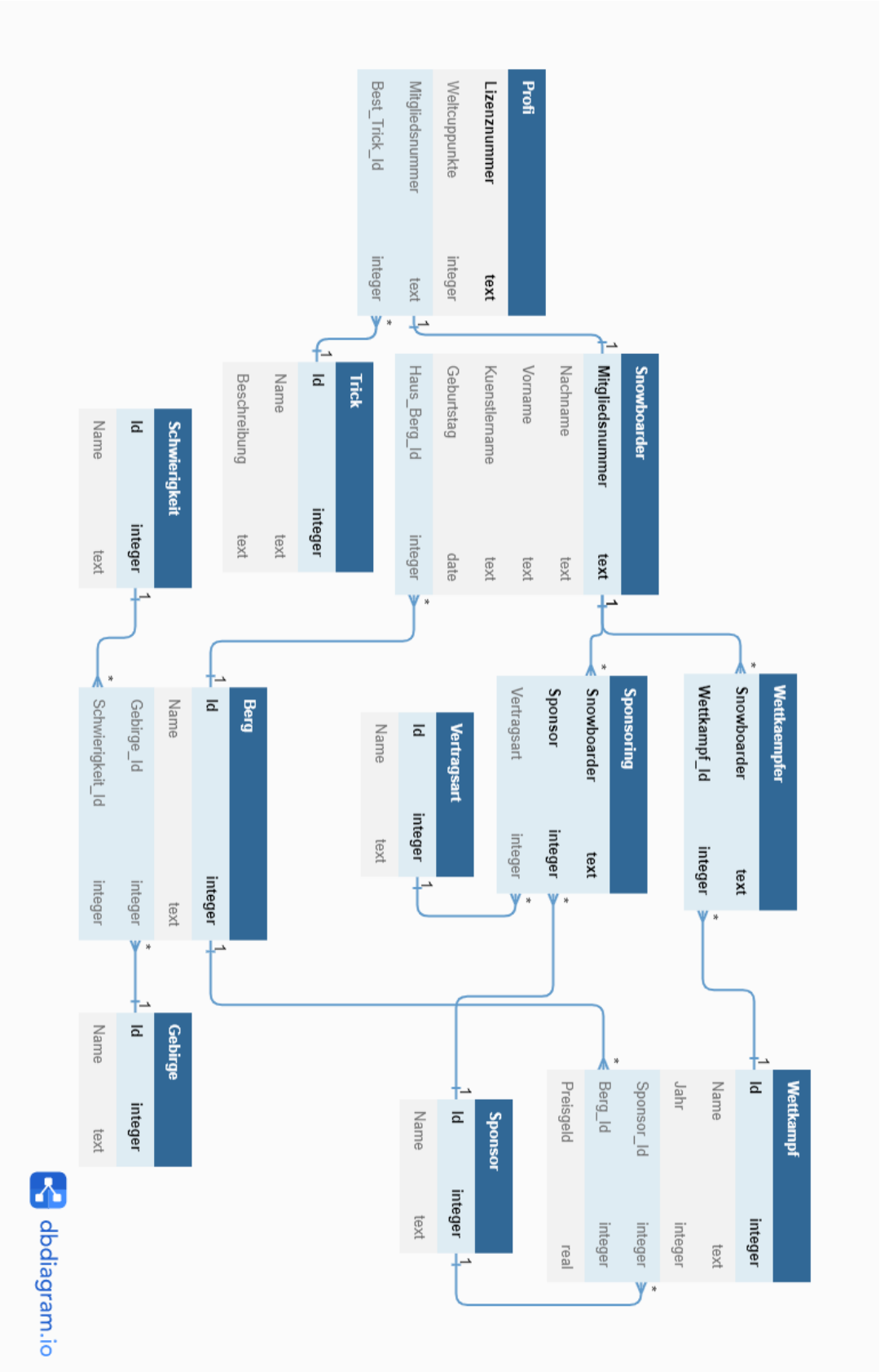
Weiterhin wurden Kommentare zu den wichtigsten Klassen im Quellcode hinterlegt.

6 Fazit

Zum Abschluss des Projekts wurden alle geforderten Funktionen erfüllt und es wird in einer Präsentation vorgestellt. Da es sich bei dem Projekt um eine schulische Leistung handelt, wird eine zukünftige Anpassung des Projekts nicht stattfinden.

A Anhang

A.1 Abbildung ER-Diagramm Datenbank



A.2 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.

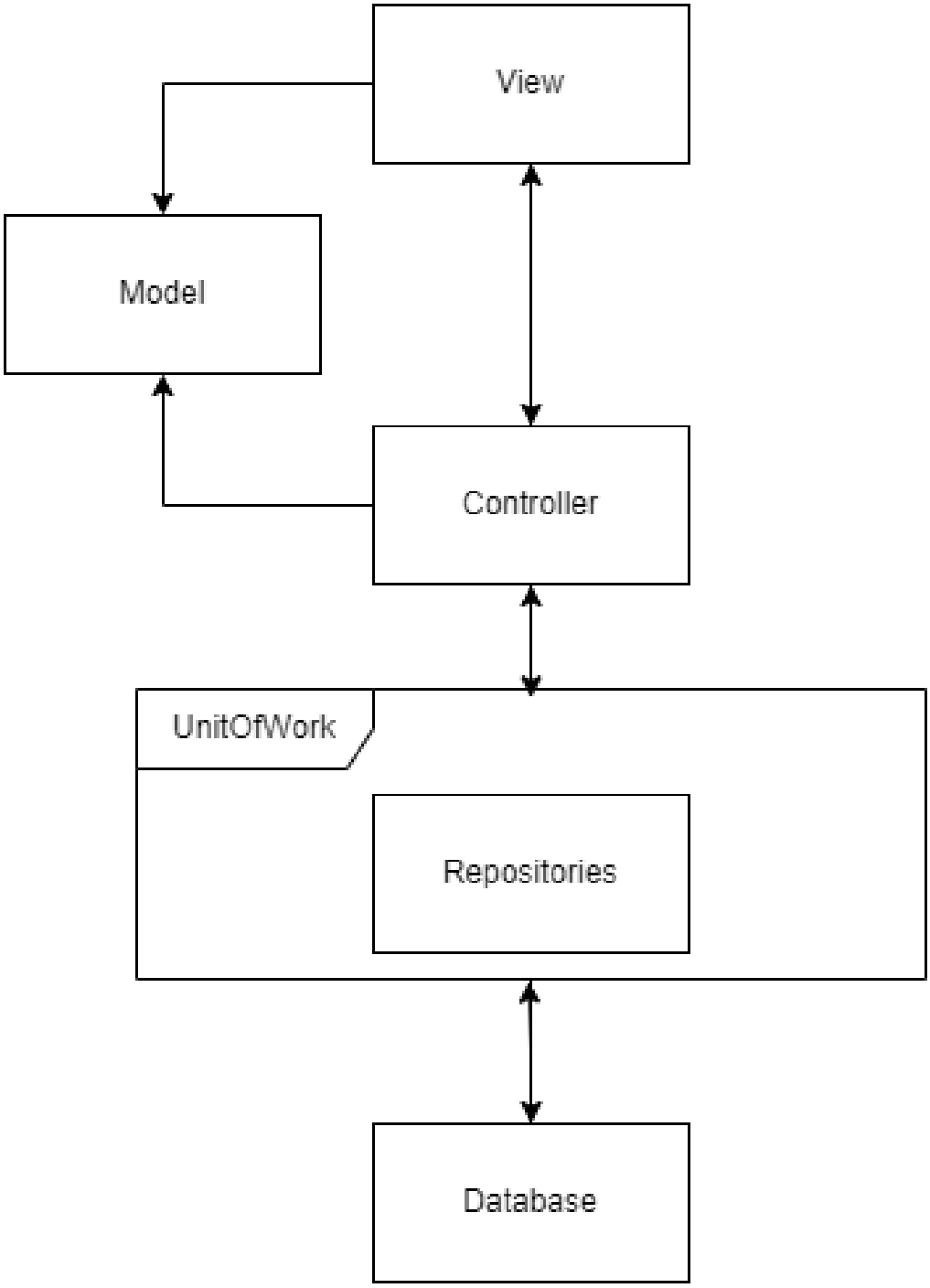
Zusätzlich soll für jeden Snowboarder der "Haus-Berg" abrufbar sein. Dieser liegt in einem Gebirge und hat eine von der ISC vergebene Schwierigkeitsstufe.

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.

Anhang

A.3 Schichten-Modell



Anhang

A.4 Klassendiagramm(Auszug)