**16.09.2020**

**Georgia antionadis, michael schneider**

**I3B**

ABSCHREIBUNG APPLIKATION

JEKATHMENAN SELVARAJAH, NIKLAUS KADERLI, IVAN FRICKER, EMIRHAN KARACA

Inhalt

[Vorwort 3](#_Toc51180523)

[Teil 1 4](#_Toc51180524)

[1.1 Vertrag zum Vorprojekt 4](#_Toc51180525)

[1.1.1 Projektname 4](#_Toc51180526)

[1.1.2 Gruppenmitglieder 4](#_Toc51180527)

[1.1.3 Betreuende Lehrpersonen 4](#_Toc51180528)

[1.1.4 Titel 4](#_Toc51180529)

[1.1.5 Beschreibung 4](#_Toc51180530)

[1.1.6 Startdatum und Abgabe 4](#_Toc51180531)

[1.1.7 Anforderungen 4](#_Toc51180532)

[1.1.8 Datum und Unterschrift der Gruppenmitglieder 4](#_Toc51180533)

[1.1.9 Datum und Unterschrift der Lehrpersonen 5](#_Toc51180534)

[1.2 Deklaration der Vorkenntnisse 5](#_Toc51180535)

[1.3 Vorarbeiten 5](#_Toc51180536)

[1.4 Benutzte Firmenstandarts 5](#_Toc51180537)

[1.5 Zeitplan 5](#_Toc51180538)

[1.6 Arbeitsjournal 7](#_Toc51180539)

[1.7 Projektorganisation 11](#_Toc51180540)

[1.7.1 Beschreibung IPERKA 11](#_Toc51180541)

[1.7.1.1 Informieren 11](#_Toc51180542)

[1.7.1.2 Planen 11](#_Toc51180543)

[1.7.1.3 Entscheiden 11](#_Toc51180544)

[1.7.1.4 Realisieren 11](#_Toc51180545)

[1.7.1.5 Kontrollieren 11](#_Toc51180546)

[1.7.1.6 Auswerten 12](#_Toc51180547)

[1.8 Organisation der Arbeitsergebnisse 13](#_Toc51180548)

[1.9 Anleitung zur Installation 13](#_Toc51180549)

[Teil 2 14](#_Toc51180550)

[2.1 Management Summary 14](#_Toc51180551)

[2.1.1 Ausgangslage 14](#_Toc51180552)

[2.1.2 Vorgehen 14](#_Toc51180553)

[2.1.3 Ergebnis 14](#_Toc51180554)

[2.2 Informieren: 14](#_Toc51180555)

[2.2.1 Beschreibung der Informationsphase 14](#_Toc51180556)

[2.2.2 Anforderungsanalyse 14](#_Toc51180557)

[2.2.3 Use-Case Diagramm 15](#_Toc51180558)

[2.3 Planen: 15](#_Toc51180559)

[2.3.1 Tätigkeitsliste 15](#_Toc51180560)

[2.3.2 GUI-Prototypen 17](#_Toc51180561)

[2.3.3 Testfallspezifikation 19](#_Toc51180563)

[2.3.4 Klassendiagramm 23](#_Toc51180564)

[2.3.5 Architektur des Programmes und der Daten 24](#_Toc51180565)

[2.4 Entscheiden 24](#_Toc51180566)

[2.4.1 Nutzwertanalyse 24](#_Toc51180567)

[2.4.2 Auswahl des GUI‘s 24](#_Toc51180568)

[2.4.3 Auswahl ob Eclipse oder Netbeans 25](#_Toc51180569)

[2.4.4 Auswahl ob MVVM oder MVC 25](#_Toc51180570)

[2.5 Realisieren 25](#_Toc51180571)

[2.5.1 Projektbeschreibung 25](#_Toc51180572)

[2.5.2 Schlüsselstellen von Programm 25](#_Toc51180573)

[2.6 Kontrollieren 27](#_Toc51180574)

[2.6.1 Testprotokoll 27](#_Toc51180575)

[2.6.2 Testbericht 27](#_Toc51180576)

[2.7 Auswerten 27](#_Toc51180577)

[2.7.1 Sitzungs-Protokolle und Resultate 27](#_Toc51180578)

[2.7.2 Persönliches Fazit 28](#_Toc51180579)

[Quellenverzeichnis 29](#_Toc51180580)

[Abbildungsverzeichnis 29](#_Toc51180581)

[Glossar 29](#_Toc51180582)

# Vorwort

Diese Dokumentation und das dazugehörige Projekt wurde von der Gruppe der IMS Baden im Kanton Aargau, bestehend aus Ivan Fricker, Emirhan Karaca, Niklaus Kaderli und Jekathmenan Selvarajah, erstellt.

„Der Aufbau der Dokumentation ist an die Vorgaben zur IPA angelehnt und orientiert sich am Dokument "Wegleitung und Weisung für Kandidaten zur „Individuellen Praktischen Arbeit“ IPA Informatiker/in EFZ und Mediamatiker/in EFZ", erstellt von den Chefexperten ICT-Berufe Aargau.“[[1]](#footnote-2)

Dieses Dokument ist in zwei Teile aufgeteilt.

Der erste Teil beinhaltet die Aufgabenstellung, die Projektorganisation, welche Projektmanagementmethode verwendet wird, der genaue Ablauf in Form eines Zeitplanes und unsere Vorkenntnisse.

Der zweite Teil beinhaltet die eigentliche Dokumentation. Darunter fällt:

* Das Informieren über das Projekt und die Aufgabe
* Die Planung
* Die Realisierung der verschiedenen Spiele und das kombinieren
* Das Testen der Anwendungen
* Die kritische Auswertung des Projektes

In diesem Projekt wurde die Projektmanagement-Methode IPERKA verwendet. Diese hat die Eigenheit, dass es als letzten Punkt «Auswerten» besitzt. Beim Auswerten können wir kritisch zurückblicken und notieren, was wir beim nächsten Mal anders machen sollten und was gut gemacht wurde.

# Teil 1

## 1.1 Vertrag zum Vorprojekt

### 1.1.1 Projektname

Vorprojekt Abschreibungen

### 1.1.2 Gruppenmitglieder

|  |  |  |
| --- | --- | --- |
| Mitglied | Klasse | E-Mail |
| Ivan Fricker | I3B | ivan.fricker@students.ksba.ch |
| Emirhan Karaca | I3B | emirhan.karaca@students.ksba.ch |
| Jekathmenan Selvarajah | I3B | jekathmenan.selvarajah@students.ksba.ch |
| Niklaus Kaderli | I3B | niklaus.kaderli@students.ksba.ch |

### 1.1.3 Betreuende Lehrpersonen

|  |  |
| --- | --- |
| Lehrpersonen | E-Mail |
| Georgia Antoniadis | Georgia.antoniadis@kanti-baden.ch |
| Michael Schneider | Michael.schneider@kanti-baden.ch |

### 1.1.4 Titel

Vorprojekt Abschreibungen

### 1.1.5 Beschreibung

Sie arbeiten als Praktikantin/Praktikant in der Buchhaltungsabteilung einer kleineren AG. Sie sehen, dass beim Jahresabschluss die Abschreibungen immer noch manuell berechnet werden. Sie schlagen der Inhaberin vor, dass Sie für die korrekte Berechnung der Abschreibungen eine Applikation programmieren (evtl. gemäss Merkblatt der ESTV für "Abschreibungen auf dem Anlagevermögen geschäftlicher Betriebe").

### 1.1.6 Startdatum und Abgabe

Startdatum des Vorprojekts: 17.08.2020

Abgabe mit Zeitpunkt: 16.09.2020, 20:00

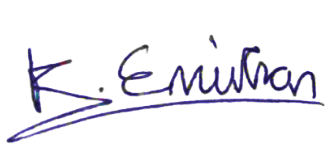
### 1.1.7 Anforderungen

Folgende Anforderungen wurden vorgängig erarbeitet:

1. Die Applikation muss folgende Eingaben entgegennehmen: Anschaffungswert, Nutzungsdauer in Jahre und Restwert (am Ende der geplanten Nutzungsdauer) oder Abschreibung in Prozenten; lineare oder degressive Aschreibung; direkte oder indirekte Abschreibung.
2. Die Applikation soll folgende Daten ausgeben: Abschreibungsbetrag pro Jahr, Buchwert (Restwert) auf Anlagekonto bzw. Buchwert auf Anlagekonto und Betrag auf Wertberichtigungskonto, eventueller Restwert nach Nutzungsdauer.
3. Jede Eingabe muss auf Plausibilität geprüft werden.
4. Bei Eingabefehlern soll der Benutzer sinnvolle Hinweise zur Korrektur erhalten.
5. Die Ausgaben sollen dem Benutzer erklärt werden.
6. Folgende Dinge müssen ***nicht*** berücksichtigt werden:
   * Buchungen beim Verkauf von Anlagen
   * Veräusserungsgewinne und Veräusserungsverluste

### Sign_IvanFricker.jpg1.1.8 Datum und Unterschrift der Gruppenmitglieder

Ivan Fricker: 24.08.2020,

Emirhan Karaca: 24.08.2020,

Jekathmenan Selvarajah: 24.08.2020,

Niklaus Kaderli: 24.08.2020,

### 1.1.9 Datum und Unterschrift der Lehrpersonen

Georgia Antoniadis: Datum, Unterschrift

Michael Schneider: Datum, Unterschrift

## 1.2 Deklaration der Vorkenntnisse

Java:

* Programmierkenntnisse aus den Module 226a, 226b und 326.
* Programmiervorkenntnisse aus den Modulen 403 und 404.

JavaFX:

* Das Arbeiten mit einem GUI aus dem Modul 120

CSS

* Vorkenntnisse aus den Modulen 101 und 120

## 1.3 Vorarbeiten

Wir hatten die Programmierwochen, welche Ähnlichkeiten mit diesem Vorprojekt besitzt. Wir mussten genauso eine Dokumentation über das ganze Projekt hinführen, wie auch eine Applikation über eine gegebene Aufgabe erstellen.

## 1.4 Benutzte Firmenstandarts

Im Projekt wurde der Firmenstandart der BBB verwendet. Das sind die aus früher erlernten Modulen. Dies bedeutet, das Erstellen von verschiedenen Modellen und Diagrammen, dass der Code einsprachig ist, das Camelcase angewendet wurde und dass der Code konstant ist.

## 1.5 Zeitplan

Folgende Daten sind für den Zeitplan gültig:

Startdatum: 17.08.2020

Enddatum: 16.09.2020

Um den Fortschritt der Arbeit zu kontrollieren wird ein Zeitplan benötigt. Der Zeitplan wurde aus den Leitfragen vom „IDPA\_Reglement\_IMS\_2020.pdf“ erstellt. Die Sollzeiten sind im Zeitplan rot markiert und die Ist-Zeiten sind grün markiert. Er wurde nach IPERKA strukturiert und in zwei Stunden Blöcken dargestellt.



Abbildung 1 Zeitplan

## 1.6 Arbeitsjournal

Im Arbeitsjournal werden die täglichen Arbeiten, aufgetretene Probleme sowie allfällige Hilfestellungen, Überzeiten und ungeplante Tätigkeiten festgehalten. Im Arbeitsjournal sind Namen nicht angegeben, da wir die Tätigkeiten als Gruppe erschaffen haben, und jeder hat mitgeholfen, so werden die Tätigkeiten nur nach Stunden aufgeteilt und nicht nach Gruppenmitglieder.

|  |  |
| --- | --- |
| Datum | 17.08.2020 |
| Ausgefühlte Arbeiten | Heute haben wir mit dem Vorprojekt angefangen. Dazu haben wir eine Aufgabe erhalten. Diese haben wir alle zuerst einmal gründlich studiert und besprochen. Zusätzlich haben wir auch noch das Reglement durchgelesen. Dann haben wir begonnen. Herr Fricker hat den Zeitplan angefangen. Herr Kaderli hat mit Scenebuilder eine grobe Skizze angefertigt. Herr Selvarajah hat die Vorlage für das MVVM-Programm erstellt und die GIT-Verbindung zwischen allen eingerichtet. Herr Karaca hat die Dokumentation angefangen. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 4, Zeitplan erstellen * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 6, GIT einrichten |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | Keine |
| Tests | Keine |
| Hilfestellung | Keine |
| Ungeplante Arbeiten | Keine |
| Überzeiten | Keine |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | Wir haben heute die Anforderungen an das Projekt erfahren. Wir konnten dann somit gut ins Projekt starten mit dem Erstellen der Dokumente und Skizzen, sowie dem Programm. |

|  |  |
| --- | --- |
| Datum | 24.08.2020 |
| Ausgefühlte Arbeiten | Heute haben wir die Arbeit von letzter Woche weitergeführt. Das heisst wir haben an der Dokumentation, am Programm, sowie der Benutzeroberfläche wurde weitergearbeitet. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 7, Dokumentation * Arbeitspaket 8, Dokumentation |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | Keine |
| Tests | Keine |
| Hilfestellung | Keine |
| Ungeplante Arbeiten | Keine |
| Überzeiten | Keine |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | Heute verlief die Arbeit gut und wird sind weit gekommen. Wir hatten keine Schwierigkeiten. |

|  |  |
| --- | --- |
| Datum | 31.08.2020 |
| Ausgefühlte Arbeiten | Heute haben wir wieder an der Programmierung und an der Dokumentation weitergemacht. Zusätzlich haben wir noch das GUI verbessert. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 7, Dokumentation |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | Keine |
| Tests | Keine |
| Hilfestellung | Keine |
| Ungeplante Arbeiten | Keine |
| Überzeiten | Keine |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | Heute verlief die Arbeit gut und wird sind weit gekommen. Wir hatten keine Schwierigkeiten. |

|  |  |
| --- | --- |
| Datum | 06.09.2020 |
| Ausgefühlte Arbeiten | Heute haben wir am Programm weitergeschrieben. Wir haben heute ausserdem die Testfallfälle angefangen zu erstellen und im Allgemeinen die Dokumentation weitergeführt. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 7, Dokumentation * Arbeitspaket 15, Abschreibungsrechner programmieren |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | * Unklarheiten, was es für Aufgaben geben soll * Fachliche Frage im Programm |
| Tests | Keine |
| Hilfestellung | Frau Antoniadis + Herrn Schneider gefragt |
| Ungeplante Arbeiten | Keine |
| Überzeiten | Herr Selvarajah hat zu Hause am Programm weitergearbeitet. |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | Diese Woche sind wir weit gekommen. Unsere Unklarheiten konnten wir dank den Lehrpersonen lösen. |

|  |  |
| --- | --- |
| Datum | 13.09.2020 |
| Ausgefühlte Arbeiten | Heute haben wir am Programm weitergeschrieben. Die Testfälle von letzter Woche haben wir fertig gestellt. Danach haben wir mithilfe von Entscheidungsmatrixen über verschiedene Optionen bestimmt. Zum Beispiel über die Darstellung der Applikation. Dazu noch die Dokumentation weitergeführt. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 7, Dokumentation * Arbeitspaket 15, Abschreibungsrechner programmieren |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | Keine |
| Tests | Keine |
| Hilfestellung | Nicht nötig |
| Ungeplante Arbeiten | Keine |
| Überzeiten | Keine |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | In dieser Woche konnten wir sehr gut arbeiten uns sind dementsprechend vorangekommen. Dabei sind keine Probleme entstanden, da wir am Anfang des Projektes eine gute Planungsphase gemacht haben. |

|  |  |
| --- | --- |
| Datum | 16.09.2020 |
| Ausgefühlte Arbeiten | In dieser Woche haben wir das Vorprojekt abgeschlossen. Das heisst wir haben letzte Kontrollen in der Dokumenation gemacht. Zudem haben wir Fehler in der Applikation getestet. |
| Erreichte Ziele | Wir konnten folgende Ziele einhalten:   * Arbeitspaket 5, Arbeitsjournal * Arbeitspaket 7, Dokumentation * Arbeitspaket 15, Abschreibungsrechner programmieren |
| Erfolge | Wir sind gut im Zeitplan |
| Misserfolge/Probleme | Keine |
| Tests | Keine |
| Hilfestellung | Nicht nötig |
| Ungeplante Abreiten | Keine |
| Überzeiten | Keine |
| Zeitplan | Wir sind gut in Zeitplan |
| Reflexion | Die Schlussphase verlief stressig, aber ohne grosse Probleme. Wir konnten uns gut organisieren in der Gruppe und konnte somit das Projekt beenden und abgeben. |

## 1.7 Projektorganisation

## 1.7.1 Beschreibung IPERKA

### 1.7.1.1 Informieren

Beim «Informieren» geht es darum, sich in die Arbeit einzulesen und die Aufgaben zu analysieren. Nach dem Analysieren geht es an das Erstellen der Anforderungsanalysen und der Use-Case Diagramme.

Folgende Leitfragen können dabei helfen:

* Wie lautet der Auftrag?
* Was sind die Vorgaben?
* Welche Infos muss ich beschaffen?
* Welche Kenntnisse muss ich mir aneignen?

### 1.7.1.2 Planen

Beim «Planen» werden für die Aufträge Lösungsvarianten vorgeschlagen. Danach werden sich Ziele und Lösungswege festgelegt und ein Zeitplan mit gut eingeteilten Arbeitsschritten erstellt. Zudem wird geprüft, welche Aufgaben zu lösen sind.

Folgende Leitfragen können dabei helfen:

* Sind die Aufgabenschritte gut aufgeteilt und priorisiert?
* Welche Aufgaben sind zu lösen?

### 1.7.1.3 Entscheiden

Im Schritt «Entscheiden» werden die Lösungsvarianten, die bei «Planen» bestimmt worden sind, verglichen und es wird sich für die beste und sinnvollste Möglichkeit entschieden. Zudem wird auch entschieden, wer was macht.

Folgende Leitfragen können dabei helfen:

* Wer macht was?
* Macht meine Idee Sinn?

### 1.7.1.4 Realisieren

Bei «Realisieren» wird mit der Umsetzung des Projektes begonnen und es wird nach Plan gearbeitet. Zudem wird dann bemerkbar, ob meine Pläne sinnvoll zum Umsetzen sind.

Folgende Leitfragen können dabei helfen:

* Bin ich gut im Zeitplan?
* Bei wem kann ich mich bei Problemen melden?
* Sind meine Pläne umgesetzt worden?

### 1.7.1.5 Kontrollieren

Beim «Kontrollieren» wird geschaut, ob das Programm fertig ist und dann auf Fehler getestet.

Folgende Leitfragen können dabei helfen:

* Habe ich mein Ziel erreicht?
* Ist das Produkt Fehlerfrei?

### 1.7.1.6 Auswerten

Bei «Auswerten» wird auf die vergangenen Projekttage kritisch zurückgeblickt und die Erfahrung wird ausgewertet.

Folgende Leitfragen können dabei helfen:

* Welche Probleme sind gelöst worden?
* Was geling mir gut?
* Was muss ich beim nächsten Mal verbessern?

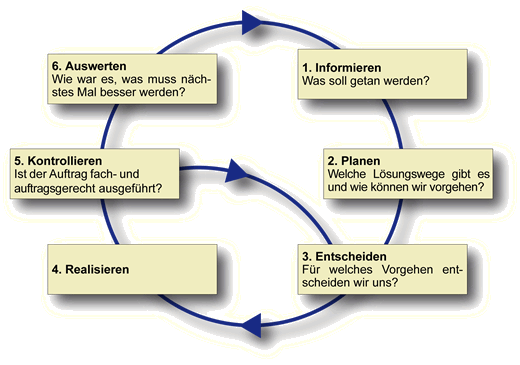


Abbildung 2 IPERKA Ablauf

## 1.8 Organisation der Arbeitsergebnisse

Die Dokumentation der Arbeit wurde täglich auf dem OneDrive von Teams gespeichert sowie gesichert und wurde von der Gruppe soeben auf OneDrive bearbeitet. Dadurch würden im Falle von Fehlern oder Löschungen keine Daten beeinflusst welche älter als ein Tag sind. Dazu konnten alle Gruppenmitglieder die Dokumentation sowie andere Daten zur gleichen Zeit sich anschauen und bearbeiten, da alles Online gespeichert ist.

Das Programm wurde soeben auch auf einem Online Speicherort gesichert, ein sogenannter GitLab-Repository. Dies Funktioniert ganz einfach und ermöglicht jedem Gruppenmitglied das Programm weiter zu entwickeln, ohne allfällige Probleme zu bekommen, und jeder hat immer das neueste Update des Programms. Jeder in der Gruppe hat ein sogenannter Branch (Veränderte Versionen) erstellt, der mit seinem eigenen Namen benennt wurde. Auf diese Branches wird dann weiterentwickelt. Wenn man am Ende des Tages das Programm sichern will, muss man sein eigenes Branch committen (oder speichern), und es mit dem Master Branch (die Hauptversion) mergen (die Hauptversion mit der veränderten Version vergleichen und zusammentun). Nachdem wird der Masterbranch gepusht (auf dem GitLab hochgeladen). Am nächsten Tag muss man das Ganze noch pullen (vom GitLab herunterladen), und dann kann man weiterentwickeln. Somit ist es möglich an jedem Tag eine Version des Codes zu erstellen, und es später herunterzuladen, damit man ein Stand des Programms nicht verlieren kann.

Die unten zusehende Liste, zeigt alle Commits (Speicherungen), die im Laufe der Programmierung hergestellt wurden. Man kann auf jeder Spielversion von jedem Tag zugreifen, und diese entsprechend herunterladen.

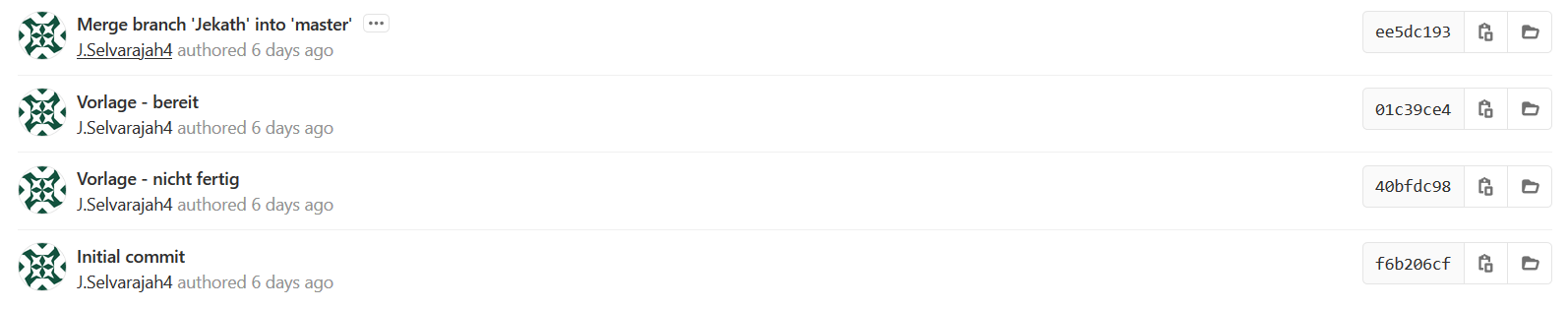


Abbildung 3 Commits auf das Repository von GitLab

Dazu mussten wir das Programm mit Maven verbinden. Maven ist ein Build-Management-Tool der Apache Software Foundation und basiert auf Java. Mit ihm kann man insbesondere Java-Programme standardisiert erstellen und verwalten. Das heisst, dass es eine Sicherung der Daten gibt, wenn die Gruppenmitglieder verschiedene Entwicklungsumgebungen benutzen.

## 1.9 Anleitung zur Installation

Zur Installation muss man zuerst Netbeans aufmachen. Danach öffnet man das Projekt (STRG + UMSCH + O). Nun startet das man das Programm einfach mit dem grünen Run Knopf.

# Teil 2

## 2.1 Management Summary

### 2.1.1 Ausgangslage

Sie arbeiten als Praktikantin/Praktikant in der Buchhaltungsabteilung einer kleineren AG. Sie sehen, dass beim Jahresabschluss die Abschreibungen immer noch manuell berechnet werden. Sie schlagen der Inhaberin vor, dass Sie für die korrekte Berechnung der Abschreibungen eine Applikation programmieren (evtl. gemäss Merkblatt der ESTV für "Abschreibungen auf dem Anlagevermögen geschäftlicher Betriebe").

### 2.1.2 Vorgehen

Das Projekt wurde mit Java, JavaFX mit Scenebuilder und CSS programmiert.

Zuerst haben wir ein MVVM-Projekt erstellt. Danach haben wir das Projekt auf Git hochgeladen und Branches für alle Gruppenmitglieder erstellt. Dann haben wir angefangen zu programmieren. Gleichzeitig haben wir Prototypen für das GUI erstellt. Anschliessend haben wir das GUI mithilfe von FXML und Scenebuilder umgesetzt. Nachfolgend haben wir die Eingabevalidierung realisiert. Zugleich haben wir die Minimize- und Exit-Funktion implementiert. Nun haben wir die Formeln ermittelt und sogleich die Berechnungen implementiert. Zuletzt haben wir die Applikation getestet und die Fehler behoben.

### 2.1.3 Ergebnis

Die Berechnung des Abschreibungsbetrags funktioniert einwandfrei. Das Ergebnis ist ein Programm mit einer grafischen Oberfläche, das Benutzereingaben liest und daraus Abschreibungen berechnet. Dabei wertet das Programm Benutzereingaben nach Fehlern aus. Bei vorhandenen Fehlern werden Warnhinweise ausgegeben.

## 2.2 Informieren:

### 2.2.1 Beschreibung der Informationsphase

In der Informationsphase haben wir uns damit befasst alle Dokumente mit den vorgegebenen Anforderungen zu studieren. Damit haben wir uns erstmal das Wissen angeeignet, um für dieses Projekt bereit zu sein.

### 2.2.2 Anforderungsanalyse

|  |  |  |
| --- | --- | --- |
| Anforderungsnummer | Anforderungstyp | Beschreibung |
| Fall-01 | Funktional | Man kann den Abschreibungsrechner schliessen. |
| Fall-02 | Funktional | Abschreibungsrechner kann minimiert werden. |
| Fall-03 | Funktional | Man kann in den Textfeldern Zahlen eingeben. |
| Fall-04 | Funktional | In den Textfeldern werden nur Zahlen akzeptiert. |
| Fall-05 | Funktional | Man kann die Lineare Abschreibung berechnen. |
| Fall-06 | Funktional | Man kann die degressive Abschreibung berechnen. |
| Fall-07 | Funktional | Man kann die Werte für die indirekte Abschreibung berechnen. |
| Fall-08 | Funktional | Man kann die Werte für die direkte Abschreibung berechnen. |
| Fall-09 | Funktional | Berechnen Button funktioniert. |
| Fall-10 | Funktional | Abbrechen Button funktioniert. |
| Fall-11 | Funktional | Ausgabe wird nach den Jahren eingeteilt. |
| Fall-12 | Funktional | Ausgaben sind korrekt. |

### 2.2.3 Use-Case Diagramm

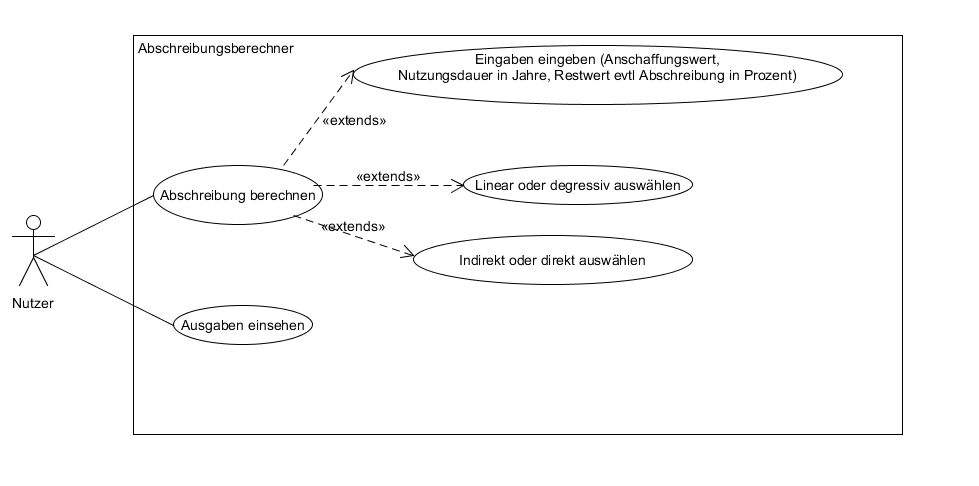


Abbildung 4 Use Case Diagramm

Der Nutzer möchte die Abschreibung für mehrere Jahren berechnen können, um das zu machen muss er die nötigen Eingaben machen können, wie zum Beispiel der Anschaffungswert, die Nutzungsdauer und mehr. Ausserdem muss er noch auswählen können ob er linear oder degressiv abschreiben will und anschliessend noch ob der die Abschreibung direkt oder indirekt verbuchen will. Die Ausgabe muss er dann auch sehen können.

## 2.3 Planen:

### 2.3.1 Tätigkeitsliste

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Startdatum | Abschlussdatum | Verantw. Person |
| Zeitplan erstellen | 24.08.2020 | 29.08.2020 | I. Fricker / N. Kaderli |
| Arbeitsjournal erstellen | 17.08.2020 | 17.08.2020 | E. Karaca / J. Selvarajah |
| Anforderungsanalyse erstellen | 17.08.2020 | 24.08.2020 | N. Kaderli | |
| Use-Case Diagramm erstellen | 29.08.2020 | 31.08.2020 | N. Kaderli |
| Tätigkeitsliste erstellen | 24.08.2020 | 24.08.2020 | I. Fricker |
| Git und Maven einrichten | 22.08.2020 | 22.08.2020 | J. Selvarajah |
| Klassendiagramm erstellen | 24.08.2020 | 29.08.2020 | N. Kaderli / J. Selvarajah |
| Skizze des Programmes erstellen | 29.08.2020 | 31.08.2020 | E. Karaca |
| Entscheidungsmatrix erstellen | 31.08.2020 | 31.08.2020 | I. Fricker | |
| Programmierung Abschreibungsrechner | 05.09.2020 | 16.09.2020 | I. Fricker / E. Karaca / N. Kaderli / J. Selvarajah |
| GUI erstellen | 31.08.2020 | 12.09.2020 | I. Fricker / E. Karaca / N. Kaderli / J. Selvarajah | |
| Dokumentation erstellen/kontrollieren | 17.08.2020 | 16.09.2020 | I. Fricker / E. Karaca / N. Kaderli / J. Selvarajah |
| Testen der Applikation | 24.08.2020 | 16.09.2020 | I. Fricker / E. Karaca / N. Kaderli / J. Selvarajah | |
| Kritische Selbstreflexion erstellen | 16.09.2020 | 16.09.2020 | I. Fricker / E. Karaca / N. Kaderli / J. Selvarajah |

### 2.3.2 GUI-Prototypen

Die grafischen Oberflächen der Programme wurden nach den ISO 9241-110 Normen erstellt.

Die Kriterien sind:

#### 2.3.2.1 Aufgabenangemessenheit

Wenn der Benutzer unterstützt wird, seine Arbeitsaufgabe effektiv und effizient zu erledigen und dabei entlastet wird.

#### 2.3.2.2 Selbstbeschreibungsfähigkeit

Wenn jeder Dialogschritt durch Rückmeldung des Dialogsystems verständlich ist oder dem Benutzer auf Anfrage erklärt wird.

#### 2.3.2.3 Steuerbarkeit

Der Benutzer besitzt die Kontrolle von Abläufen oder er bekommt Wahlmöglichkeiten.

#### 2.3.2.4 Erwartungskonformität

Wenn es Konsistent ist und den Merkmalen des Benutzers entspricht.

#### 2.3.2.5 Fehlertoleranz

Falsche oder fehlerhafte Eingaben des Benutzers erkennen und ihm die Möglichkeit zur Korrektur geben.

#### 2.3.2.6 Individualisierbarkeit

Wenn es die Vorlieben des Benutzers zulässt und zur Erfüllung der Aufgabe Arbeitserleichterung in Form von individualisierbaren Dialogen und Ergebnissen anbietet.

#### 2.3.2.7 Lernförderlichkeit

Wenn es dem Benutzer beim Erlernen unterstützt und anleitet.

#### 2.3.2.8 Mock Abschreibungsrechner

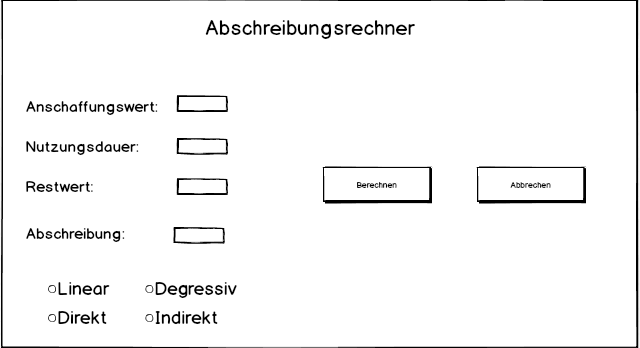


Abbildung 5 Mock der ersten Seite des Programms

## 

Abbildung 6 Mock der zweiten Seite des Programms

**Implementierung der ISO Normen:**

1. Aufgabenangemessenheit

Das Programm besitzt alle Funktionen, sodass man es benutzen kann. Der Benutzer hat immer die Übersicht und kann seine Arbeit leicht erledigen.

2. Selbstbeschreibungsfähigkeit

Die Knöpfe sind klar nach ihrer Funktion beschriftet und der Benutzer erhält eine Rückmeldung, was gerade beim Druck passiert ist.

3 Steuerbarkeit

Der Benutzer kann auf die Abläufe, auf die er Zugriff hat, ausführen und diese werden nach seinen Vorgaben erfüllt.

4 Erwartungskonformität

Die Darstellung ist simpel und konsistent und für den Benutzer ist es.

5 Fehlertoleranz

Bei falschen Eingaben in den Eingabefeldern wird der Benutzer auf den Fehler mit einer Meldung hingewiesen.

6 Individualisierbarkeit

Je nachdem was der Benutzer an Eingaben macht, so kommen auch immer andere Zahlen heraus auf der nächsten Seite.

7 Lernförderlichkeit

Das Spiel leitet und unterstützt den Spieler mit Texten und Knöpfen bei seiner Nutzung des Programms.

#### 2.3.2.9 GUI Abschreibungsrechner

Abbildung 7 GUI der ersten Seite des Programms

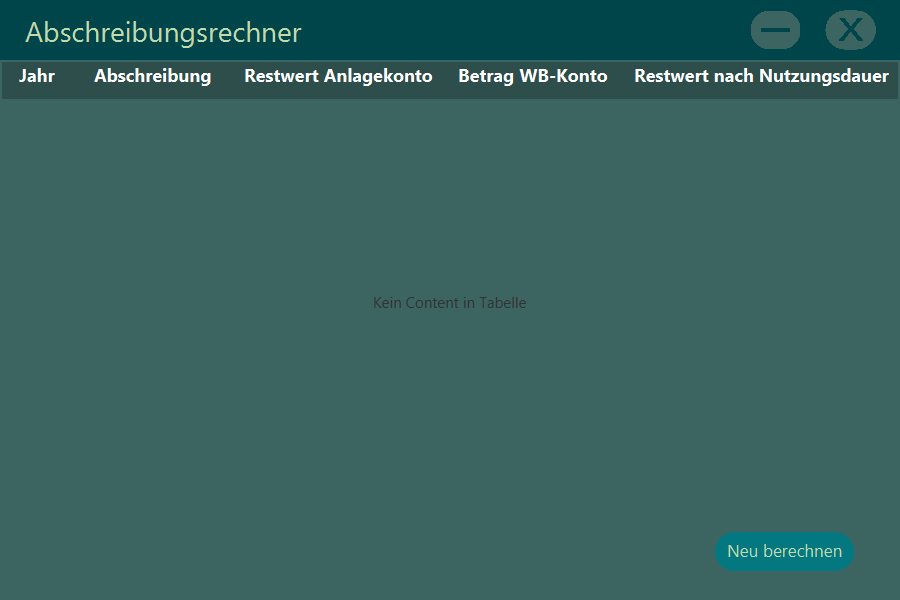


Abbildung 8 GUI der zweiten Seite des Programms

### 2.3.3 Testfallspezifikation

|  |  |
| --- | --- |
| Testfallnummer: | Fall-01 |
| Anforderung: | Man kann den Abschreibungsrechner schliessen. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Oben rechts auf das «X» klicken |
| Ausgabe: | 1. Der Rechner schliesst sich |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-02 |
| Anforderung: | Abschreibungsrechner kann minimiert werden |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Oben rechts auf das «-» klicken |
| Ausgabe: | 1. Das Fenster wird minimiert. |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-03 |
| Anforderung: | Man kann in den Textfeldern Zahlen eingeben |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Ins Feld “Anschaffungswert” eine Zahl eingeben mit dem Format (XXX.XX CHF) 2. Beim Feld «Nutzungsdauer» eine ganze Zahl eingeben 3. Beim Feld «Restwert» eine Zahl eingeben mit dem Format (XXX.XX CHF) 4. Beim Feld “Abschreibung in Prozent”/ “Abschreibungsbetrag“ eine Zahl eingeben mit dem Format (XXX.XX CHF) |
| Ausgabe: | 1. Die Werte stehen in den Textfeldern 2. Mit dem Wert kann gerechnet werden |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-04 |
| Anforderung: | In den Textfeldern werden nur Zahlen akzeptiert. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Bei “Nutzungsdauer” abc eingeben |
| Ausgabe: | 1. Unterhalb vom Feld mit dem falschen Wert wird ein Warnhinweis angezeigt |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-05 |
| Anforderung: | Man kann die lineare Abschreibung berechnen. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Auf linear klicken 2. Auf Berechnen klicken |
| Ausgabe: | 1. Radiobutton is markiert 2. Hinweis Restwert und Abschreibung in Prozent erscheinen 3. Abschreibung wird linear berechnet |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-06 |
| Anforderung: | Man kann die degressive Abschreibung berechnen. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Degressiv auswählen 2. Berechnen klicken |
| Ausgabe: | 1. Radiobutton is markiert 2. Abschreibung wird degressiv berechnet |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-07 |
| Anforderung: | Man kann die Werte für die indirekte Abschreibung berechnen. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Indirekt auswählen 2. Berechnen klicken |
| Ausgabe: | 1. Abschreibungsbetrag wird im «Betrag-WB Konto» ausgegeben |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-08 |
| Anforderung: | Man kann die Werte für die direkte Abschreibung berechnen. |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Direkt auswählen 2. Berechnen klicken |
| Ausgabe: | 1. Abschreibungsbetrag wird im «Restwert Anlagekonto» ausgegeben |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-09 |
| Anforderung: | Berechnen Button funktioniert |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Zahlen eingeben wie beim «Fall-03» 2. Linear oder degressiv auswählen 3. Direkt oder indirekt auswählen 4. Auf «Berechnen» klicken |
| Ausgabe: | 1. Ausgabe kommt |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-10 |
| Anforderung: | Abbrechen Button funktioniert |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Zahlen eingeben wie beim «Fall-03» 2. Linear oder degressiv auswählen 3. Direkt oder indirekt auswählen 4. Auf «Abbrechen» klicken |
| Ausgabe: | 1. Eingaben werden aus den Feldern entfernt 2. Auswahl mit den Radiobuttons werden entfernt |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-11 |
| Anforderung: | Ausgabe wird nach den Jahren eingeteilt |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Schritte 1-4 beim Fall-08 2. Bei «Nutzungsdauer» 10 Jahre eintragen |
| Ausgabe: | 1. Ausgabe für jedes der 10 Jahre wird ausgegeben Jahr 1 zuoberst Jahr 10 zuunterst. |

|  |  |
| --- | --- |
| Testfallnummer: | Fall-12 |
| Anforderung: | Ausgaben sind korrekt |
| Voraussetzungen: | Abschreibungsrechner ist offen |
| Eingabe: | 1. Bei «Anschaffungswert» 100 eingeben 2. Bei «Nutzungsdauer» 5 Jahre eintragen 3. Linear auswählen 4. Direkt auswählen |
| Ausgabe: | 1. Ausgabe für 5 Jahre 2. Im Jahr 1 beim Restwert für Anlagekonto 80, im Jahr 2 60 usw. 3. Bei Abschreibung steht immer 20 4. Bei Restwert nach Nutzungsdauer steht 0 |

### 2.3.4 Klassendiagramm

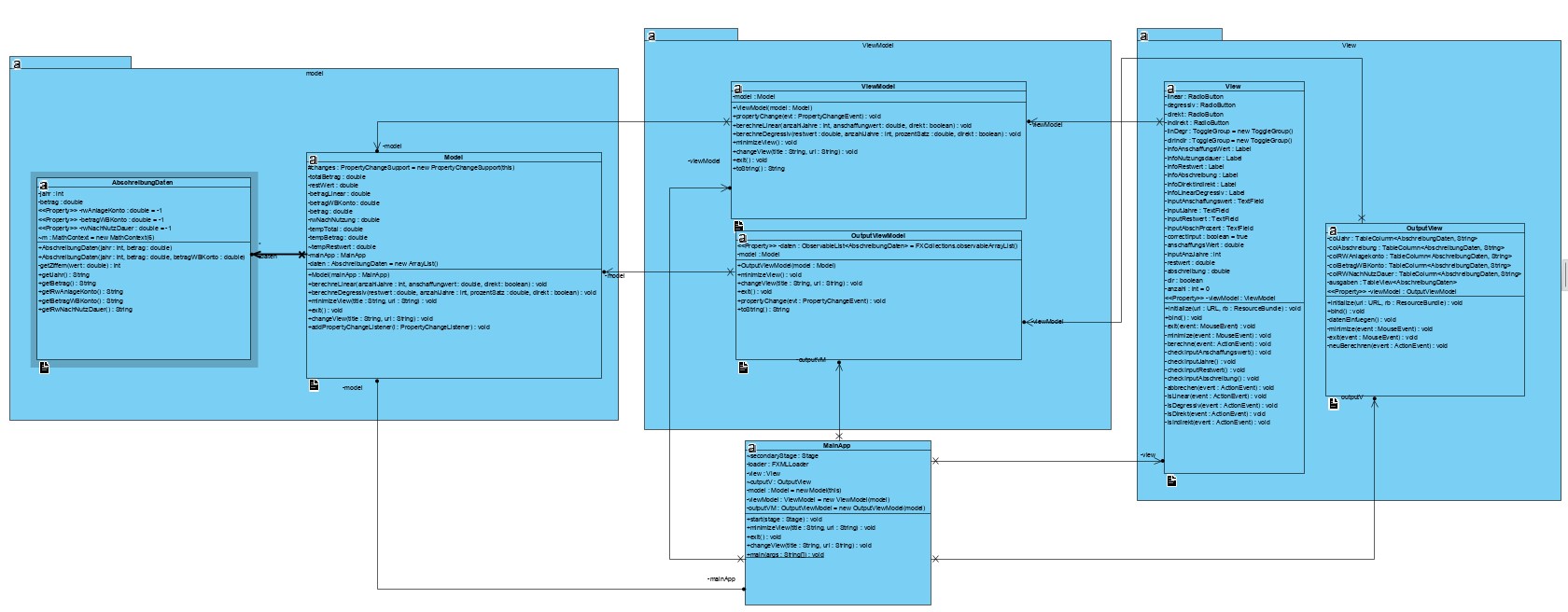


Abbildung 9 Klassendiagramm

### 2.3.5 Architektur des Programmes und der Daten

Die Architektur des Programms ist MVVM. Das steht für Model, View Viewmodel. Das Model repräsentiert die Businesslogik und ist zuständig für die Speicherung von Daten. Das Viewmodel bereitet die Daten für die Ausgabe vor. Die View zeigt die Daten mittels GUI an.

## 

## 2.4 Entscheiden

### 2.4.1 Nutzwertanalyse

Die Nutzwertanalyse hilft bei der Auswahl von Alternativen! Sie eignet sich besonders zur Bewertung von verschiedenen Handlungsalternativen und dient somit als Entscheidungshilfe. Die Methode kann immer dann eingesetzt werden, wenn die Bewertung von Alternativen nicht in erster Linie anhand von konkreten Zahlen und Fakten geschieht, sondern auch subjektive Einflüsse.

Die Nutzwertanalyse wird in folgenden Schritten durchgeführt:

1. Festlegung der Alternativen bzw. Entscheidungsvarianten

Die verschiedenen Varianten, aus denen eine ausgewählt werden soll, werden gesammelt und festgehalten. Es ist hilfreich, wenn diese Liste nicht allzu lang wird, weil die Methode sonst sehr aufwändig wird.

1. Definition von Bewertungskriterien

Es werden die Kriterien festgelegt, anhand derer eine Entscheidung getroffen werden soll. Diese Kriterien sind häufig Anforderungen an das Produkt oder zu erreichende Ziele. Hier entsteht meist eine Liste von bis zu 10 Kriterien.

1. Gewichtung der Bewertungskriterien

Jedem Kriterium wird ein Prozentsatz hinterlegt, der die Wichtigkeit des Kriteriums belegt. Die Summe der Einzelgewichtungen muss 100% ergeben.

1. Festlegung des Bewertungsmaßstabes

Die einzelnen Kriterien werden mit Punkten bewertet. Um hier eine Eindeutigkeit sicherzustellen, muss der Bewertungsmaßstab genau definiert werden, z.B. 5 Punkte = sehr gut, 1 Punkt = mangelhaft.

1. Bewertung der Alternativen

Hier erfolgt die eigentliche Bewertung: Pro Kriterium und Alternative werden nun Punkte vergeben und die gewichteten Punkte berechnet.

1. Summierung und Auswahl

Durch Summierung der Einzelgewichtungen ergibt sich die gewichtete Punktzahl pro Alternative. Die Alternative mit der höchsten Punktzahl entspricht den definierten Kriterien am besten.

### 2.4.2 Auswahl des GUI‘s

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Benutzerfreundlich | \*4 | Lesbarkeit | \*5 | Einfachere Bedingung | \*3 | Schönheit des GUI | \*3 | Total |
| Grünbasierte GUI | 4 | 16 | 5 | 25 | 3 | 9 | 4 | 12 | 62 |
| Rotbasierte GUI | 4 | 16 | 3 | 15 | 4 | 12 | 3 | 9 | 52 |
| Schwarzbasierte GUI | 4 | 16 | 1 | 5 | 4 | 12 | 2 | 6 | 39 |

Maximale Bewertung: 5

Minimale Bewertung: 1

### 2.4.3 Auswahl ob Eclipse oder Netbeans

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Nutzen für Projekt | \*3 | Einfachheit | \*2 | Vorkenntnisse | \*3 | Total |
| Eclipse | 2 | 6 | 4 | 8 | 1 | 3 | 17 |
| Netbeans | 4 | 12 | 3 | 6 | 4 | 12 | 30 |

Maximale Bewertung: 5

Minimale Bewertung: 1

### 2.4.4 Auswahl ob MVVM oder MVC

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Nutzen für Projekt | \*2 | Einfachheit | \*2 | Vorkenntnisse | \*6 | Total |
| MVC | 3 | 6 | 4 | 8 | 3 | 18 | 32 |
| MVVM | 4 | 8 | 4 | 8 | 5 | 30 | 46 |

Maximale Bewertung: 5

Minimale Bewertung: 1

Wir entschieden uns für MVVM, da wir uns damit besser auskannten. Auch fanden wir die Verbindung einfacher und effizienter.

## 2.5 Realisieren

### 2.5.1 Projektbeschreibung

1. Die Applikation kann folgende Eingaben entgegennehmen: Anschaffungswert, Nutzungsdauer in Jahre und Restwert (am Ende der geplanten Nutzungsdauer) oder Abschreibung in Prozenten; lineare oder degressive Abschreibung; direkte oder indirekte Abschreibung.
2. Die Applikation kann folgende Daten ausgeben: Abschreibungsbetrag pro Jahr, Buchwert (Restwert) auf Anlagekonto bzw. Buchwert auf Anlagekonto und Betrag auf Wertberichtigungskonto, eventueller Restwert nach Nutzungsdauer.
3. Jede Eingabe wird auf Plausibilität geprüft.
4. Bei Eingabefehlern kann der Benutzer sinnvolle Hinweise zur Korrektur erhalten.

### 2.5.2 Schlüsselstellen von Programm

Ein Bild, das Screenshot enthält.

Automatisch generierte BeschreibungDie Schlüsselstellen vom Programm sind Model und die View. Im Model gibt es zwei Methoden berechneLinear und berechneDegressiv. Diese Methoden sind zuständig für die Berechnung. Nach der Berechnung speichern Sie die Werte in einer ArrayList und rufen PropertyChangeSupports firePropertyChanges mit der List auf. In der View werden die festgelegten Methoden je nach Aktion (Knopfdruck, Auswahl Checkbox) automatisch aufgerufen.

Abbildung 10 Erste Schlüsselstelle

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 11 Zweite Schlüsselstelle

## 2.6 Kontrollieren

### 2.6.1 Testprotokoll

Testumgebung

OS: Windows 10 Home / Pro

JRE: Version 8.2

Internetverbindung: Wired, Standard zum Teil Wireless

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Resultat | Datum | Tester | Bemerkungen |
| 1 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 2 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 3 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 4 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 5 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 6 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 7 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 8 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 9 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 10 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 11 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |
| 12 | Erfüllt | 16.09.2020 | J. Selvarajah | Keine |

### 2.6.2 Testbericht

Die Applikation erfüllt alle Testfälle und funktioniert fehlerfrei. Die Applikation ist bereit für die Freigabe. Wir sind auch sehr zufrieden mit den Testergebnissen.

## 2.7 Auswerten

### 2.7.1 Sitzungs-Protokolle und Resultate

**Eröffnung:**

Die regelmässige Sitzung von IDPA Gruppe 7 wurde ordnungsgemäss einberufen und fand jeweils am Montag ab den 16.08.2020 an der Kantonschule statt.

**Anwesend waren:**

Emirhan Karaca, Ivan Fricker, Jekathmenan Selvarajah und Niklaus Kaderli

1. **Dokumentation**

Bei der Dokumentation haben wir geschaut, dass jeder gleich viel machen muss und das Resultat ist vollständig.

1. **Programm**

Auch beim Programmieren versuchten wir es gleich aufzuteilen und das Resultat hat funktioniert.

1. **Präsentation**

Die Präsentation werden wir in der nächsten Sitzung besprechen.

**Abschluss:**

Die nächste Sitzung findet am 21.09.2020 an der Kantonsschule statt.

Protokoll verfasst von: Ivan Fricker

Genehmigt von: Emirhan Karaca, Ivan Fricker, Jekathmenan Selvarajah und Niklaus Kaderli

### 2.7.2 Persönliches Fazit

#### Fazit von Ivan Fricker

Die Interdisziplinäre Projektarbeit hat mir sehr viel Spass gemacht und ich habe auch viel gelernt. Allgemein verstehe ich das Programmieren besser als zuvor. Auch die Dokumentation hat mir lehrreiche Dinge für die Zukunft gebracht. Die Gruppenarbeit war auch sehr gut, wir haben das meiste zusammen gemacht und meistens gut kommuniziert.

Ich hatte mehrere Probleme, erstens wusste ich nicht genau was der Unterschied von der Tätigkeitsliste und dem Zeitplan ist. Meine Gruppenmitglieder wussten es auch nicht und ich fragte die Lehrpersonen. Auch hatte ich manchmal Schwierigkeiten beim Programmieren und musste mir Hilfe von der Gruppe holen.

Allgemein war die Projektarbeit sehr lehrreich und erfolgreich.

#### Fazit von Emirhan Karaca

Dieses Vorprojekt hat mir selbst vieles beigebracht und mich für das kommende Hauptprojekt vorbereitet. Dadurch, dass wir die Programmierwochen im letzten Frühling hatten, fiel mir persönlich die Bearbeitung dieses Projektes viel leichter, da einiges als Repetition galt. Zudem habe ich erneut das Arbeiten in einer Gruppe repetiert, welches mir sicher auch in der Zukunft in weiteren Projekten hilft.

In diesem Projekt verlief die Gruppenarbeit sehr gut. Jeder wusste, war er zu tun hatte, und machte dementsprechend seine Arbeit ohne jegliche Macken.

Das einzige, grössere Problem war das Verstehen der Anforderungen an die Applikation und die Dokumentation.

Ansonsten lief das Projekt sehr gut und machte Spass.

#### Fazit von Jekathmenan Selvarajah

Dieses Projekt ist meiner Meinung nach, ein grosser Erfolg. Unsere Teamarbeit war gut gelungen und unser Programm funktioniert einwandfrei. Wir konnten gut miteinander kommunizieren und die Arbeiten sehr gut aufteilen. Bei Problemen konnten wir uns gegenseitig aufeinander verlassen. Probleme hatte ich beim Programm. Ich musste einen ganzen Tag lang nichts anderes als Fehlersuche betreiben, dabei war es eher ein kleiner Fehler.

#### Fazit von Niklaus Kaderli

In diesem Vorprojekt konnte ich einige nützliche Sachen mitnehmen für das kommende Projekt. Aufgrund der Programmierwoche im letzten Frühling war aber vieles was wir gemacht haben vor allem in der Dokumentation nur Repetition. Die Teamarbeit hat sehr gut funktioniert, wir konnten die Aufgaben gut verteilen damit jeder immer etwas zu tun hatte.

Das einzige grössere Problem in meinen Augen war die Analyse der Anforderungen für die Dokumentation zum Beispiel welche Dokumente es beinhalten musste. Ebenfalls brauchten wir zu Beginn ein bisschen zu viel Zeit mit dem Verstehen der Aufgabestellung für das Programm.

Im grossen Ganzen konnten wir aber das Vorprojekt gut umsetzen und hatten auch zeitlich nicht allzu grosse Probleme.

# Quellenverzeichnis

UML

<https://www.computerweekly.com/de/definition/Unified-Modeling-Language-UML>

JavaFX <https://de.wikipedia.org/wiki/JavaFX#:~:text=JavaFX%20ist%20ein%20Framework%20zur%20Erstellung%20plattform%C3%BCbergreifender%20Java%2DApplikationen.>

Stackoverflow:

<https://stackoverflow.com/>

MVVM:

<https://de.wikipedia.org/wiki/Model_View_ViewModel>

MVC:

<https://de.wikipedia.org/wiki/Model_View_Controller>

Abschreibungen:

<https://de.wikipedia.org/wiki/Abschreibung#:~:text=Abschreibung%20(englisch%20depreciation%2Famortization),Gegensatz%20ist%20die%20Zuschreibung.>

# Abbildungsverzeichnis

[Abbildung 1 Zeitplan 6](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181001)

[Abbildung 2 IPERKA Ablauf 12](#_Toc51181002)

[Abbildung 3 Commits auf das Repository von GitLab 13](#_Toc51181003)

[Abbildung 4 Use Case Diagramm 15](#_Toc51181004)

[Abbildung 5 Mock der ersten Seite des Programms 17](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181005)

[Abbildung 6 Mock der zweiten Seite des Programms 18](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181006)

[Abbildung 7 GUI der ersten Seite des Programms 19](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181007)

[Abbildung 8 GUI der zweiten Seite des Programms 19](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181008)

[Abbildung 9 Klassendiagramm 23](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181009)

[Abbildung 10 Erste Schlüsselstelle 27](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181010)

[Abbildung 11 Zweite Schlüsselstelle 28](https://kantonsschulebaden-my.sharepoint.com/personal/ivan_fricker_students_ksba_ch/Documents/Microsoft%20Teams-Chatdateien/Dokumentation_Abschreibung.docx#_Toc51181011)

# 

# Glossar

**Branch**

Ein Branch repräsentiert eine unabhängige Entwicklungslinie. Er dient als Abstrahierung für den Edit/Stage/Commit-Prozess. Am besten können wir uns einen Branch als eine brandneue Kombination aus Arbeitsverzeichnis, Staging-Bereich und Projekthistorie vorstellen. Neue Commits werden in der Historie des aktuellen Branchs festgehalten, was in einem Fork in der Historie des Projekts resultiert.

**Commit**

Sobald an einer Working Copy gearbeitet wird protokolliert Git alle getätigten Änderungen mit. Mittels commit können die Änderungen zu dem Repository hinzugefügt werden, eine neue Version der Datei(n) befinden sich dann im Repository.

**Git**

Git wird für die Versionierung von Dateien eingesetzt. Vor allem im Programmierbereich dient es dazu die eigenen Änderungen zu überwachen, sie rückgängig zu machen, sie anderen über sogenannte "Repositories" (Repos) zur Verfügung zu stellen oder Aktualisierungen von anderen einzuholen. Im Gegensatz zu SVN besitzt Git kein zentrales Repository, sondern tritt als verteiltes System auf - aus der Sicht eines Bearbeiters ist die eigene Arbeitskopie (Working Copy) ein eigenes Repo für sich. Bei Bedarf können Änderungen von öffentlich zugänglichen Repos in die eigene Working Copy eingespielt werden oder die eigenen Änderungen z.B. über einen Patch den anderen zugänglich gemacht werden.

**GitLab**

Ist eine von der BBB zur Verfügung gestellte und verwaltete Version von GitHub.

**JavaFX**

JavaFX ist ein Framework zur Erstellung plattformübergreifender Java-Applikationen. Es ist eine Java-Spezifikation von Oracle und setzt sich zum Ziel, das professionelle Erstellen und Verteilen von interaktiven, multimedialen Inhalten und grafischen Benutzeroberflächen (GUIs) über sämtliche Java-Plattformen hinweg zu erleichtern.

**Merge**

Mit Git merge werden Änderungen aus einem anderen Branch in den aktiveBranch übernommen. Dabei werden die Änderungen, die zwischenzeitlich in dem zu mergenden Branch erfolgt sind, auf den aktuellen Stand im aktiven Branch angewendet.

**Model**

Das Modell enthält Daten, die von der Präsentation dargestellt werden. Es ist von Präsentation und Steuerung unabhängig. Die Änderungen der Daten werden der Präsentation durch das Entwurfsmuster „Beobachter“ bekanntgegeben.

**MVVM**

Model View ViewModell (MVVM) ist eine Variante des Model-View-Controller (MVC)-Konzeptes zur Trennung von Layout und Programmcode. Dabei erhält die in XAML gestaltete Benutzeroberfläche (View) alle Daten durch Datenbindung und der Programmcode erhält alle Aktionen der Benutzeroberfläche mit Hilfe sogenannter Commands. MVVM realisiert eine klare Kompetenztrennung und eine lose Kopplung zwischen Benutzerschnittstelle und Benutzerschnittstellensteuerung.

**Pull**

Pull-Requests erleichtern Entwicklern die Zusammenarbeit. Die Entwickler können Änderungsvorschläge über eine benutzerfreundliche Weboberfläche diskutieren, bevor sie in das offizielle Projekt eingearbeitet werden. Mit der einfachsten Form von Pull-Requests können Entwickler anderen Teammitgliedern mitteilen, dass sie ein Feature fertiggestellt haben. Sobald ein Feature-Branch fertig ist, übermittelt der Entwickler einen Pull-Request. So werden alle Beteiligten darüber informiert, dass sie den Code prüfen und in den master-Branch mergen müssen.

**Push**

Der Befehl git push wird verwendet, um Inhalte aus einem lokalen Repository in ein Remote-Repository hochzuladen. Per Push überträgst du Commits aus deinem lokalen Repository in ein Remote-Repository. Der Befehl ist das Gegenstück zu git fetch: Ein Fetch importiert Commits in lokale Branches, ein Push exportiert Commits in Remote-Branches.

**UML**

Die Unified Modeling Language (UML), im Deutschen auch vereinheitlichte Modellierungssprache, ist eine grafische Standard-Notation für die Modellierung von Objekten der realen Welt. Sie wird insbesondere zur Spezifikation, Konstruktion und Dokumentation von Softwareteilen und anderen Systemen eingesetzt.

**View**

Die Präsentation ist für die Darstellung der Daten des Modells und die Realisierung der Benutzerinteraktionen zuständig. Sie kennt das Modell, dessen Daten sie präsentiert, ist aber nicht für die Verarbeitung dieser Daten zuständig. Des Weiteren ist sie von der Steuerung unabhängig.

**ViewModel**

Eine Klasse, die beim MVVM-Pattern verwendet wird. Dient als eine Verbindung zwischen der View und dem Model.

1. Text von IDPA\_Reglement\_IMS\_2020.pdf [↑](#footnote-ref-2)