Universität Regensburg
Fakultät für Wirtschaftswissenschaften
Professur für Wirtschaftsinformatik - Prof. Dr. Guido Schryen

Entwicklung eines Zeitplan-Generators für Leichtathletikveranstaltungen

Projektseminar

Eingereicht bei: Prof. Dr. Guido Schryen

Eingereicht am 11. Februar 2016

Eingereicht von:

Thomas Baumer, Benedikt Bruckner
E-Mail Adresse: Thomas.Baumer@stud.uni-regensburg.de, Benedikt Bruckner@stud.uni-regensburg.de
Matrikelnummer: 1721141, 1728475

Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

Bsp.	Beispiel
SaaS	Software as a Service

Kapitel 1

Einleitung

Das richtige Zeitmanagement ist für jeden eine Herausforderung, da es oft nicht einfach ist alle Termine in einer begrenzten Zeit abzuarbeiten. Deshalb ist es wichtig Termine bzw. Veranstaltungen zu planen. Auch für die Organisation von Leichtathletikveranstaltungen bedarf es einer strukturierten und vor allem richtigen Zeitplanung, um Unannehmlichkeiten zu vermeiden bzw. einen reibungslosen und professionellen Ablauf der Veranstaltung zu gewährleisten. Momentan werden die Zeitpläne der Veranstaltungen der LG Telis Finanz manuell aus einem Ausgangsfile generiert. Da das Dokument mit den relevanten Daten sehr unstrukturiert ist und viele überflüssige Informationen enthält, ist der Zeitplan natürlich sehr aufwändig zu erstellen. Zudem ist bei einer manuellen Erstellung die Möglichkeit eines fehlerhaften Zeitplans immer gegeben.

Im Rahmen des Projektseminars „Erstellung eines Zeitplan-Generators für Leichtathletikveranstaltungen“ soll deshalb ein Tool entwickelt werden, welches ermöglicht, dass aus html-Ausgangsfiles einer bestimmten Form automatisch ein strukturierter Zeitplan in html-Form erstellt wird. Das Tool erkennt alle benötigten Daten in dem Ausgangsfile, die für die Erstellung des Zeitplans notwendig sind und generiert daraus eine Tabelle. Dabei sollen das Datum der Veranstaltung, die Uhrzeit des Wettkampfbeginns, die Altersklasse (offizielle Bezeichnung) mit dem zugehörigen Geschlecht und die Wettkampfart (mit allgemeingültigen Abkürzungen) Bestandteil der gewünschten Ergebnistabelle sein. Ziel ist es für jeden Wettkampfart eine separate Tabelle zu erzeugen, weshalb zuerst nach dem Datum sortiert werden muss. Gleichzeitig soll das Datum als Überschrift für die jeweiligen Zeitpläne aufgenommen werden. Der Zeitplan selbst soll folgendermaßen aufgebaut sein: Der Beginn der Veranstaltung steht in der ersten Spalte unter dem Titel Zeit, wobei aufsteigend nach der Uhrzeit sortiert wird; die Altersklassen werden in der ersten Zeile ab der zweiten Zelle angeführt, wobei diese aszendierend und nach dem Geschlecht sortiert dargestellt werden und die Wettkampfbezeichnung wird schließlich in der richtigen Zelle ausgehend von Datum, Uhrzeit und Altersklasse eingetragen. Für die Ergebnistabelle dient als Referenz der Zeitplan der Sparkassen Gala 2016. Die Herausforderung besteht in der Übersichtlichkeit, Korrektheit und Kompaktheit der Tabelle.

Die automatische Generierung eines Zeitplans für Leichtathletikveranstaltungen hat viele positive Auswirkungen, wie beispielsweise der geringere Zeitaufwand für den Er-

steller. Es ist nicht mehr nötig mühsam alle relevanten Informationen zu identifizieren und diese anschließend in den Zeitplan zu übertragen, da dies der Zeitplan-Generator komplett übernimmt. Der Nutzer muss lediglich das gewünschte Ausgangsfile auswählen, um einen fertigen und übersichtlichen Zeitplan zu erhalten. Des Weiteren ermöglicht der Zeitplan-Generator eine einheitliche Darstellung von Leichtathletikzeitplänen. Alle Terminpläne für die jeweiligen Veranstaltungen werden nach dem gleichen Schema erzeugt, was es für den Nutzer natürlich auch leichter macht sich zurecht zu finden. Bei der Erstellung werden immer die offiziellen Altersklassen bzw. Kategorien der Leichtathletikwettbewerbe berücksichtigt. Zudem muss sich der Nutzer nicht bei jeder neuen Veranstaltung auf unterschiedliche Namenskonventionen bzw. eine andere Tabellendarstellung umstellen, da die Daten immer gleich strukturiert sind und beispielsweise keine Teilnehmerklassen zusammengefasst werden. Dies führt natürlich wieder dazu, dass unnötige Missverständnisse vermieden werden und die Benutzerfreundlichkeit erhöht wird. Ein weiterer Vorteil bei der automatischen Erstellung eines Zeitplans ist, dass das Tool vielseitig verwendet werden kann. Als Input wird lediglich ein html-File benötigt, dass in einer bestimmten Form die relevanten Daten für die Terminplanerstellung enthält. Dies macht es beispielsweise auch für kleinere Verbände leicht möglich einen Terminplan zu generieren, der online verfügbar ist und den allgemeinen Standards entspricht. Ein weiterer motivierender Grund für das Projekt ist die Richtigkeit und Konsistenz der Daten bei einer automatischen Generierung. Das Tool erkennt alle relevanten Daten maschinell, was einen Datenverlust ausschließt und einen vollständigen Zeitplan garantiert. Die Fehlerquelle beim manuellen Erstellen des Terminplans wird also eliminiert, da beispielsweise keine Tippfehler oder falsche bzw. fehlende Eintragungen mehr möglich sind. Dieser Punkt ist für die Leichtathleten bzw. alle Interessierten für Leichtathletikveranstaltungen von essentieller Bedeutung, da aufgrund falscher Terminplaninformationen die Anreise, das Training und der Tagesablauf beeinträchtigt werden. Diese Punkte werden natürlich an den Termin der Veranstaltung ausgerichtet und im schlimmsten Fall hätte das zur Folge, dass der Sportler zu seinem Wettkampf nicht antreten kann, da er aufgrund einer falschen Information zu spät oder gar nicht zum Wettkampfort angereist ist. Das würde natürlich ein schlechtes Licht auf die Veranstaltung bzw. den Veranstalter werfen.

Es gibt also viele gute Gründe, die die Entwicklung eines Zeitplan-Generators für Leichtathletikveranstaltungen rechtfertigen. Besonders die Strukturiertheit und Richtigkeit sprechen für die Umsetzung des Projektes. Im Folgenden wird nun das genaue Vorgehen bei der Entwicklung des Zeitplan-Generators eingegangen.

Kapitel 2

Hauptteil

2.1 Grundlegende Überlegungen zur Durchführung des Projektes

Das Projektseminar „Erstellung eines Zeitplan-Generators für Leichtathletikveranstaltungen“ wird von Thomas Baumer und Benedikt Bruckner (beide 5. Semester Wirtschaftsinformatik) unter Betreuung von Gerit Wagner durchgeführt. Als erstes stellte sich die Frage wie das Projekt umgesetzt werden soll und welche Software bzw. Programmiersprache zur Problemlösung verwendet werden sollen. Für die Wahl einer Programmiersprache zwischen JavaScript und Python konnte schnell eine Entscheidung getroffen werden, obwohl JavaScript und Python beides Sprachen sind für die eine Vielzahl an Dokumentationen, Bibliotheken und Tutorials existieren. Zudem wird ein klares und übersichtliches Programmieren bei beiden ermöglicht. Da Thomas Baumer und Benedikt Bruckner aber bereits im Kurs „Internettechnologien und Network-Computing“ erste Erfahrungen in JavaScript sammeln konnten, entschied man sich für JavaScript und sparte sich so eine größere Einarbeitungszeit in eine neue Programmiersprache. Mit der Verwendung von JavaScript können alle gewünschten Ziele erreicht werden. Des Weiteren wurde die Verwendung einer Versionsverwaltungssoftware diskutiert. Da eine Versionsverwaltung viele Vorteile bietet, wie beispielsweise das Wiederherstellen von alten Zuständen des Projekts und die Protokollierung, wo jede Änderung an den Dateien mit Autor und Datum, also die ganze Versionsgeschichte, nachvollzogen werden kann, entschied man sich auch schnell und eindeutig für die Möglichkeit einer Versionsverwaltung. Außerdem ist es viel leichter, nachvollziehbarer und übersichtlicher als beispielsweise das Verschicken von einzelnen Codeteilen per E-Mail an die anderen Projektteilnehmer. (Quelle: PdP-Skript) Zudem bietet eine Versionsverwaltung noch die Möglichkeit Zugriffe und Entwicklungszweige zu koordinieren, was jedoch bei diesem Projekt aufgrund der Komplexität hinsichtlich der Projektteilnehmer nicht zwingend notwendig ist. Als Versionsverwaltungssoftware einigte man sich schließlich auf den „GitHub Desktop“, da man schon im Kurs „Praxis des Programmierens“ positive Erfahrungen mit Git sammeln konnte. Als Editor zur Erstellung von html- bzw. JavaScript-Code wurde einerseits unter Windows Notepad++ und

andererseits für Mac die Software Brackets verwendet, da diese Open Source-Editoren sind und viele Sprachen, wie beispielsweise JavaScript, jQuery und html unterstützen. Sie genügen also allen im Projekt erwartenden Ansprüchen. Bei der Entwicklung wurde vordergründig der Browser Google Chrome verwendet, da dieser alle verwendeten Sprachen, Frameworks und Methoden unterstützt. Ein weiteres Ziel ist eine fertige Desktop-App zu entwickeln. Dafür macht man sich die Software node.js zunutze. Der Vorteil hier ist, dass nach dem Erstellen des eigentlichen Programms mit dem Editor nur noch kleine Anpassungen vorgenommen werden müssen, um ein fertiges Programm zu erhalten. Dies hat die positive Auswirkung, dass der Nutzer nicht extra die verschiedenen html-Dateien abspeichern muss, womit er sich möglicherweise auch nicht sehr gut auskennt. Ein eigenständiges Programm erleichtert die Bedienbarkeit und die Nutzerfreundlichkeit. Darüber hinaus diskutierte man zu diesem frühen Stadium des Projektes bereits wie die abschließende Projektseminararbeit geschrieben werden soll. Betreuer Gerit Wagner empfahl die Verwendung des Softwarepaketes LaTex, das bei der Erstellung von größeren Abschlussarbeiten viele Vorteile gegenüber Microsoft Office Word bietet, abgesehen von dem Nachteil der Einarbeitungszeit in die ungewohnte Umgebung. Ein Argument ist, dass LaTex die Formatierung und den Inhalt voneinander trennt, indem man zu verändernde Textstellen mit Befehlen in einem Editor kennzeichnet, was ein sauberes und genau festlegbares Layout zur Folge hat. (https://de.wikipedia.org/wiki/LaTeX) Außerdem gibt es seitens des Lehrstuhls bereits eine LaTex-Vorlage für Projektseminararbeiten auf die zurückgegriffen werden kann. Somit können die strengen Anforderungen für die Formatierung bzw. Gestaltung einer umfangreichen Projektseminarabschlussarbeit durch ein sauberes Layout erzielt werden. Zudem gibt es bereits grafische Editoren die den Umgang mit LaTex vereinfachen. Im Rahmen des Projekts wurde die Software TeXworks verwendet. Des Weiteren kann man für LaTex-Dokumente wieder ein Git-Repository anlegen und so alle Vorteile einer Versionsverwaltung nutzen.

2.2 Konzeptionelle Phase und Identifikation möglicher Probleme bzw. Schlüsselstellen

Dieser Unterpunkt soll einen Überblick über die allgemeine Vorgehensweise geben, die später noch detaillierter erläutert wird. Nach ersten Überlegungen mit welchen Mitteln das Projekt umgesetzt werden soll, folgten viele Gedanken über mögliche Schwierigkeiten bzw. die Herangehensweise bei der Erstellung des Zeitplan-Generators. Zuerst wurde natürlich das Ausgangsfile genauer betrachtet. Es wurde festgestellt, dass das Ausgangsfile sehr viele irrelevante Daten beinhaltet und man jeweils nur die zwei Zeilen über den Teilnehmerlisten benötigt. Parallel dazu verglich man die Zusammensetzung der Daten mit den dafür vorgesehenen Positionen in der Ergebnistabelle. Als Referenz für die Ergebnistabelle wurde der Zeitplan der Sparkassen Gala 2016 verwendet. Es stellte sich heraus, dass die Aufschlüsselung der relevanten Daten eine Schwierigkeit werden könnte, da die Datensätze teilweise unterschiedliche Strukturen haben und die Informationen

oft anhand mehrerer Suchkriterien identifiziert werden müssen. Des Weiteren wurde deutlich, dass die Befüllung der Tabelle eine Herausforderung werden kann, da die Wettkämpfe anhand des Datums, der Uhrzeit und der Teilnehmerklasse zugeordnet werden müssen. Bei der Umsetzung muss also ein besonderes Augenmerk auf die Identifikation der relevanten Daten und die saubere Befüllung der Ergebnistabelle gelegt werden. Ausgehend der gemeinsamen Überlegungen im Team wurden fünf größere Schritte für die Problemlösung bzw. Erstellung des Zeitplan-Generators definiert: Der erste Schritt ist die Spezifikation des Rohtextes. In diesem Prozess wird die genau Form des Ausgangsfiles festgehalten, was insbesondere den Inhalt und die Struktur beinhaltet. Dabei ist es von besonderer Wichtigkeit zu erkennen an welchen Positionen die wesentlichen Daten, wie Datum, Uhrzeit, Teilnehmerklasse und Wettkampfbezeichnung stehen. Der darauffolgende Schritt ist das Einlesen des Ausgangsfiles. Um mit dem Rohtext zu arbeiten muss dieser natürlich zuerst vom Programm identifiziert und eingelesen werden. Hier ist es die Aufgabe des Nutzers das gewünschte Ausgangsfile, das die zuvor spezifizierte Form aufweist, auszuwählen. Der dritte Schritt ist, dass man aus dem Ausgangsfile nur die relevanten Daten herausfiltert. Hier sollen praktisch nur noch alle Absätze angezeigt werden. Anschließend müssen diese Daten im vierten Schritt gesäubert werden. Das bedeutet, dass anhand bestimmter Suchkriterien das Datum, die Uhrzeit, die Teilnehmerklassen und die Wettkampfbezeichnung erkannt werden. Im letzten Schritt wird schließlich der gewünschte Zeitplan aus den zuvor identifizierten Informationen generiert. Die Tabelle soll den gleichen Charakter wie die Referenztabelle der Sparkassen Gala 2016 haben. Kopie aus Einleitung bei Problemdefinition umformulieren bzw. anpassen Für jeden Wettkampftag soll dabei eine eigene Tabelle erzeugt werden. Als Überschrift für eine Tabelle dient das Datum mit dem zugehörigen Wochentag. Der Beginn der Veranstaltung steht in der ersten Spalte unter dem Titel Zeit, wobei aufsteigend nach der Uhrzeit sortiert wird; die Altersklassen werden in der ersten Zeile ab der zweiten Zelle angeführt, wobei diese aszendierend und nach dem Geschlecht sortiert dargestellt werden und die Wettkampfbezeichnung wird schließlich in der richtigen Zelle ausgehend von Datum, Uhrzeit und Altersklasse eingetragen. Zusammenfassend werden also neben dem vorhandenen Ausgangsfile vier html-Files erstellt, die der Problemlösung dienen. Dies hat den Vorteil der sauberen Trennung der unterschiedlichen Funktionen und des weiteren ist es bei der Programmierung leichter umzusetzen. Zudem ist eine gute Testbarkeit der Funktionalität bzw. einfacheres Debugging ermöglicht. Die genaue Vorgehensweise in den einzelnen Schritten wird nun im Folgenden genauer erläutert.

2.3 Spezifikation des Rohtextes

Die erste Anforderung an den Rohtext ist, dass dieser das Dateiformat html besitzen soll. Dies ermöglicht eine genaue Navigation mit Hilfe des DOM (in Abkürzungsverzeichnis vermerken). Im Folgenden wird nun der Inhalt und die Struktur des Ausgangsfiles näher spezifiziert. Grundsätzlich kann der Rohtext beliebigen html-Code beinhalten mit der

Ausnahme des im später erläuterten Aufbau eines Paragraphen. Das für die Entwicklung verwendete Ausgangsfile weist beispielsweise folgenden Aufbau auf. In den ersten zwei Zeilen des Dokumentes steht die Überschrift („Laufnacht und Sparkassen Gala 2016“) und der Ort bzw. das Datum („Regensburg, von 04.06.2016-05.06.2016“) geschrieben. Daraufhin folgen verschiedene Verweise, wie die Altersklassen (z.B. „Männer“ und „weibliche Jugend U20“) und alle Wettkampfbezeichnungen (z.B. „100m (Vorprogramm) - Zeitläufe“). Im Anschluss kommen die relevanten Zeilen für die Erstellung des Zeitplanes. Im Testfile beispielsweise: - Zeile 1: „100m (Vorprogramm), Frauen + U20 + U18 – Zeitläufe“ - Zeile 2: „Datum: 05.05.2016 Beginn: 12:00“ Diese zwei Zeilen sind DOM-Elemente der Form ... . Es handelt sich hier also um Paragraphen und man kann alle relevanten Daten mit Hilfe dieser Form identifizieren. Der Paragraph ist im Allgemeinen also folgendermaßen aufgebaut: Zeile 1 beinhaltet die Teilnehmerklassen (z.B. „Frauen + U20 + U18“) ungefähr in der Mitte der Zeile. im String gekennzeichnet folgt auf die ersten Zeichen und ein Komma (xxxx,) die Teilnehmerklasse. Auf die Teilnehmerklasse folgt ein Bindestrich Minuszeichen? und die nachfolgenden Zeichen (-xxxx). Die Teilnehmerklassen werden also initiiert durch ein Komma und beendet durch einen Bindestrich. Zudem enthalten alle Teilnehmerklassen die Zeichenfolgen „weiblich“, „weibliche“ oder „Frauen“ bzw. „männlich“, „männliche“ oder „Männer“. Außerdem ist es möglich, dass in diesem Abschnitt mehrere Teilnehmerklassen, abgetrennt durch ein „+“, enthalten sind. Auf das Pluszeichen würde dann ein U<Zahl><Zahl> folgen. Ein weiterer Bestandteil der ersten Zeile ist der Inhalt der runden Klammern und die runden Klammern selbst. Dort befinden sich Zusatzinformationen, die entfernt werden, wenn es sich um „Hauptprogramm“, „Gala“, „Vorprogramm“ oder „Laufnacht“ handeln sollte. Ansonsten bleibt der Inhalt der Klammern an dieser Stelle stehen. Es gibt aber auch den Fall, dass keine Klammern enthalten sind. Den Rest der Zeichenkette (nach Entfernung der Teilnehmerklasse und gegebenenfalls der Klammern) bildet die Disziplin. Die Disziplin setzt sich also aus den ersten Zeichen bis zur Klammer und den letzten Zeichen nach dem Bindestrich zusammen. Im Beispiel entsteht die Disziplin „100m Zeitläufe“. Die zweite Zeile enthält das Datum und die Uhrzeit. Der erste Eintrag in der zweiten Zeile ist „Datum“ gefolgt vom Datum selbst in der Form dd.mm.yyyy (Abkürzungsverzeichnis). Der zweite Eintrag in der zweiten Zeile ist „Beginn“ gefolgt von der Uhrzeit des Wettkampfstarts mit der Form hh:mm.

Abgesehen davon kommt nach den Paragraphen im verwendeten Ausgangsfile immer eine Tabelle mit der Startnummer, dem Namen, dem Jahrgang, der Nationalität, dem Verein, der Saisonbestleistung und der persönlichen Bestleistung, sowie den verschiedenen Einträgen der Wettkampfteilnehmer. Dies ist jedoch für die Zeitplangenerierung nicht weiter relevant.

Zusammenfassend muss das Ausgangsfile also im html-Format vorliegen und DOM-Elemente mit der Form ... enthalten, die als Inhalt die relevanten Daten haben. Zudem müssen die Paragraphen den wie oben beschriebenen Aufbau aufweisen, damit alle Daten im Anschluss richtig verarbeitet werden können. Nachdem

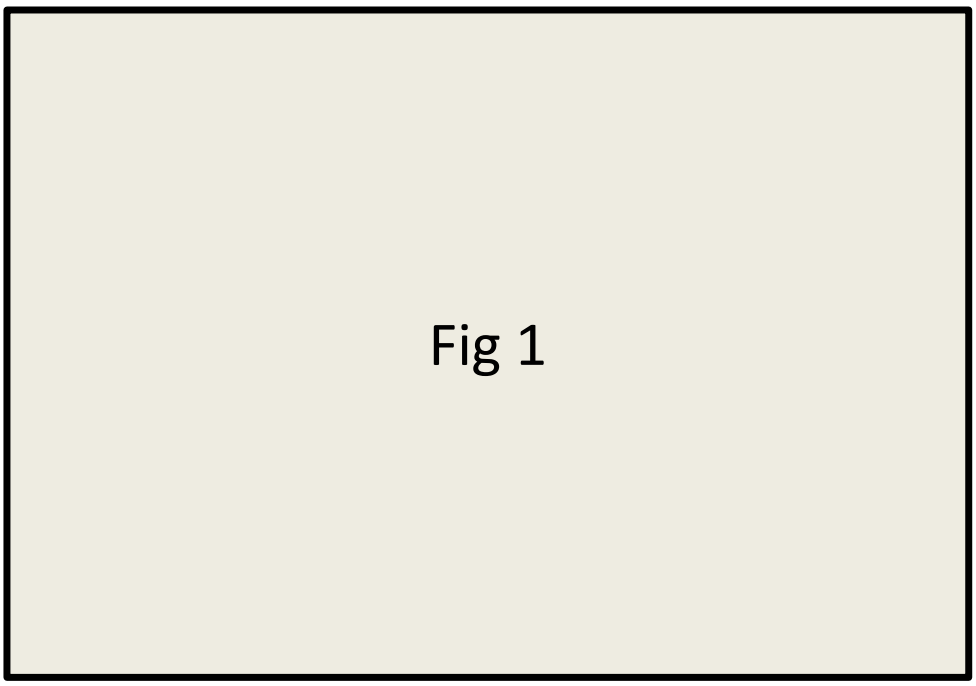


Abbildung 2.1: Beispielgrafik

der Rohtext genauer spezifiziert wurde, wird die Funktionsweise des File-Uploaders im Folgenden genauer erklärt.

Ausgangsfile evtl. an Arbeit anhängen

2.4 Beispiele

Dies ist eine Referenz auf ein Paper [**?**]. Die Verwaltung der Referenzen erfolgt in der Datei References.bib. Zur Bearbeitung der Referenzen kann beispielsweise das Programm JabRef[1] verwendet werden.

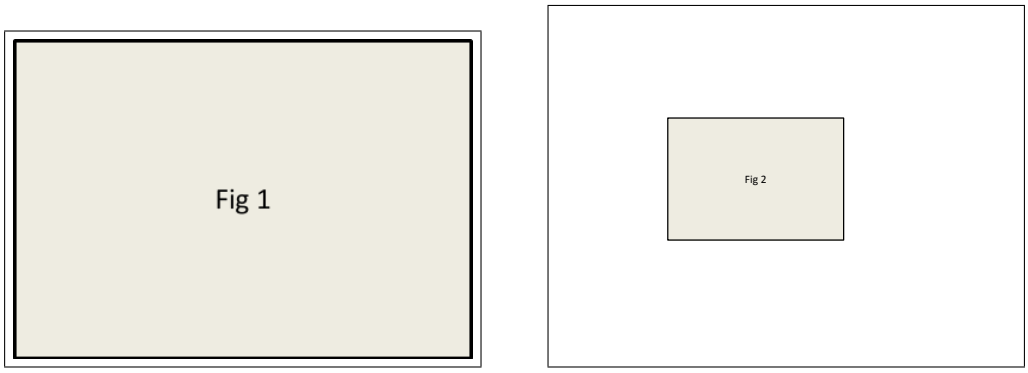
Besonders interessant ist auch die automatische Erstellung des Abkürzungsverzeichnisses. Zuerst wird die Abkürzung definiert um bei erstmaliger Verwendung im Abkürzungsverzeichnis zu erscheinen: Beispiel (Bsp.), Software as a Service (SaaS)

Referenzen auf Grafiken: 2.1, 2.2(b), 2.2

```
1 class HalloWorld {
2   public static void main(String[] args) {
3   String message="Hallo World!";
4   System.out.println(message);
5   }
6 }
```

Listing 2.1: HelloWorld

[1] http://jabref.sourceforge.net/



(a) subfigure 1 (b) subfigure 2

Abbildung 2.2: Beispiel subfigure

Kapitel 3

Schluss

Anhang

Anhang A

Erster Anhang

Anhang B

Zweiter Anhang

B.1 Anhang

B.2 Anhang

Erklärung an Eides statt

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Regensburg, den 11. Februar 2016

Thomas Baumer, Benedikt Bruckner
Matrikelnummer 1721141, 1728475