Praxissemester bei isys vision GmbH & Co. KG

Bericht und Erfahrungen

Lorenz Bung lorenz.bung@htwg-konstanz.de
HTWG Konstanz
Angewandte Informatik, 4. Semester

9. April 2018

Zusammenfassung

In diesem Bericht geht es um mein Praktikum bei isys vision GmbH & Co. KG, welches ich im Rahmen meines Studiums im Fach Angewandte Informatik im 4. Studiensemester absolviert habe.

Ich werde beschreiben und erklären, welche Aufgaben mir weshalb gegeben wurden, wie ich sie gelöst habe und welche Werkzeuge und Vorgehensweisen ich dabei verwendet habe. Weiterhin werde ich die firmeninternen Abläufe, Ziele und Methoden nennen.

Inhaltsverzeichnis

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

0. Einleitung

Zu Beginn möchte ich kurz die Hintergründe meines Praktikums erläutern, z.B. den Bewerbungsprozess, die Wahl der Firma und einige Informationen zu isys vision.

Isys vision ist ein Softwarehersteller, der sich mit grafischen Lösungen auseinandersetzt. Die Bildverarbeitung steht hierbei im Vordergrund und wird durch die von isys produzierte Software durchgeführt. Die im selben Gebäude gelegene Tochterfirma Ensenso stellt Hardwarekomponenten her, welche dann in Kombination mit der Software von isys vision als Gesamtpaket an Endkunden, aber auch Großabnehmer verkauft wird. Dies umfasst hauptsächlich Komponenten wie Kameras und Beleuchtungsbauteile sowie Verbindungskabel usw.

Schon im Grundstudium war mir klar, dass ich für mein praktisches Studiensemester Firmen im Bereich der Bildverarbeitung, -generierung oder aber im Webumfeld suchen werde. Mit einem Praktikum in genannten Gebieten wollte ich einerseits meine Entscheidung für die Vertiefungsrichtung der Medieninformatik im Hauptstudium bekräftigen, andererseits war dies schon immer ein Bereich der Informatik, welcher mich besonders gereizt hat. Aus diesen Gründen habe ich mich unter Anderem bei isys vision beworben und schlussendlich ein halbes Jahr hier verbracht.

1. Web-Oberfläche für das IVS

Das erste größere Projekt war die Entwicklung einer Web-Oberfläche für ein bereits bestehendes, jedoch noch nicht vollständiges Bildverarbeitungssystem. Sowohl Webinterface als auch Bildverarbeitung waren bereits vorhanden, jedoch nicht für den aktuellen Anwendungszweck geeignet und somit war eine Überarbeitung unabdingbar.

Die Aufgabe des betreffenden Systems ist die Ausrichtung einzelner Keramikseiten, welche dann in weiteren Schritten verarbeitet werden. Dabei geht es um die Produktion von Lambdasonden, wie sie in der Abgasfilterung von Autos zum Einsatz kommen.

Diese Keramik-"sheets" werden dabei mit hochpräziser Genauigkeit ausgestanzt, weswegen sie vor dem Stanzen entsprechend ausgerichtet werden müssen. Eine physikalische Ausrichtung ist dabei so gut wie unmöglich, da die gewünschte Genauigkeit dabei nicht erreicht werden kann.

Das von isys entwickelte System erkennt nun mithilfe zweier Kameras die Lage zweier Marken auf dem Sheet und somit die Lage auf dem verschieb- und rotierbaren Tisch. Mithilfe der Bildverarbeitungssoftware wird nun eine Motorbewegung errechnet, welche eine Ausrichtung des Werkstücks bewirkt. Anschließend kann das Teil durch weitere Maschinen in seiner nun festgelegten Position weiterverarbeitet werden.

Die Software besteht dabei nun aus drei verschiedenen Komponenten:

- Dem *ScbDrv*, welcher für die Ansteuerung der Motoren und Hardware verantwortlich ist.
- Dem *IVS* (Isys Vision Server), welcher für die Auswertung der Kameradaten sowie die Bildverarbeitung und Errechnung der Endposition verantwortlich ist.
- Dem *Webinterface*, welches dem Endkunden eine Möglichkeit zur Einsicht des aktuellen Status sowie zur Kontrolle über die Maschine gibt.

Während der ScbDrv nur geringfügig an die neue Maschinenkonfiguration angepasst werden musste, war mein Betreuer mit der Anpassung des IVS an die neuen Marken (in diesem Fall die Ecken der Keramiksheets) beschäftigt. Meine Aufgabe bestand aus dem visuellen Redesign der Weboberfläche, dem Anpassen des Inhaltes (Entfernen überflüssiger Elemente, Hinzufügen von neuen, wichtigen Dingen) sowie der Anpassung der Funktionalität mithilfe von HTML, CSS und JavaScript.

1.1 Einlesen in vorhandenen Code und Redesign

Da das Interface auf der alten, bereits vorhandenen Version aufbauen sollte, war das Einarbeiten in bereits vorhandenen Code notwendig. Um Änderungen vorzunehmen war dabei ein großes Verständnis für diesen Code nötig, da es sich um komplexe und vor allem abstrakte Abläufe handelte, die nicht leicht nachzuvollziehen waren.

1.1.1 Flat-Design mithilfe von CSS

Da meine bisherigen Erfahrungen mit Javascript und jQuery mittelmäßig bis wenig vorhanden waren, machte ich mir zunächst das grafische Redesign der Webseite zur Aufgabe. Mit CSS hatte ich bereits gearbeitet, sodass ich mich verhältnismäßig schnell in das vorhandene Stylesheet einarbeiten konnte und erste Änderungen vornahm.

Dazu gehörten beispielsweise das Entfernen von Schatten (`box-shadow`), was Buttons und Menüs deutlich besser lesbar und sauberer macht, oder die Anpassung von Farben (hellgrau wird ersetzt durch weiß etc). Auch wurde ein Hintergrundbild von mir entfernt und durch eine durchgängige Farbe ersetzt, was zum gewünschten "cleanen" Erscheinungsbild beitrug.

Durch bereits wenige Änderungen im Stylesheet konnte ich hierbei recht große Erfolge erzielen, was die Modernität der Seite angeht. Mit diesen Grundlagen mussten nun Kleinigkeiten angepasst werden, wie beispielsweise zu große Listeneinträge in der Menüleiste, fehlender Umrandung von Buttons oder Ähnlichem. Diese Anpassungen waren minimal, jedoch war der Zeitaufwand dafür immens, da Regeln im CSS immer wieder unterdrückt oder überschrieben wurden und erst das richtige Vorgehen bzw. die dafür verantwortliche Regel ermittelt werden musste.

Schlussendlich waren die Anpassungen jedoch vorgenommen und ich konnte mich auf Basis eines sauberen, nutzbaren und leserlichen Designs dem Verständnis und der Modifikation des Javascript-Frameworks widmen.

1.1.2 Javascript-Framework

In der bereits vorhandenen Version der Webseite wurde sämtliche Funktionalität durch ein Javascript-Framework geregelt, welches unabhängig von Inhalt und Design steht. Im Folgenden ging es nun darum, den hier geschriebenen Code durchzugehen, nachzuvollziehen, zu verinnerlichen und zu verstehen. In etwa 1500 Zeilen waren Initialisierungsfunktionen, Kommunikationsmodule und Eventhandler für Buttons etc. undokumentiert aufzufinden.

Aufgrund der trotz weniger Codezeilen doch sehr hohen Komplexität und vor allem den fehlenden Kommentaren gestaltete es sich als sehr schwierig, sich einzuarbeiten. Ein weiterer Nachteil war meine mangelnde Erfahrung mit Javascript; durch viel Recherche und weiterführende Beispiele konnte ich mir jedoch einiges herleiten und somit verstehen.

Meine erste Änderung war hierbei das Ersetzen von javascript-basierten Zoom-Buttons durch eine Leiste (in HTML ein `<input range>`-Element). Dies sah zum einen deutlich besser aus und war einfacher zu bedienen, außerdem war somit der Inhalt (in diesem Fall die Zoom-Leiste im HTML) besser von der Funktion getrennt.

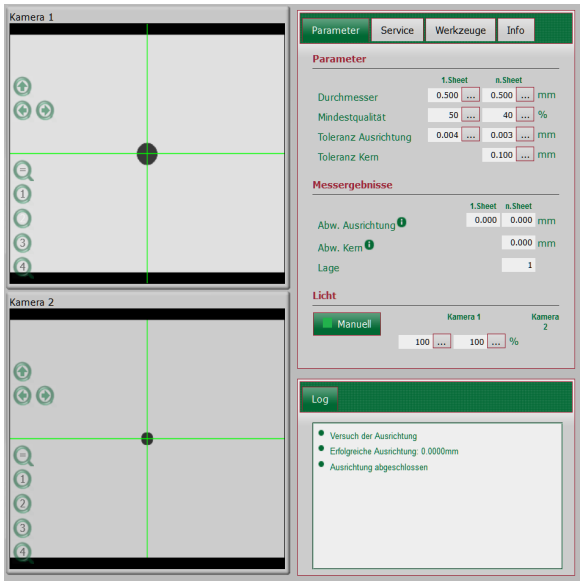

Abbildung 1.1: Interface vor dem Redesign

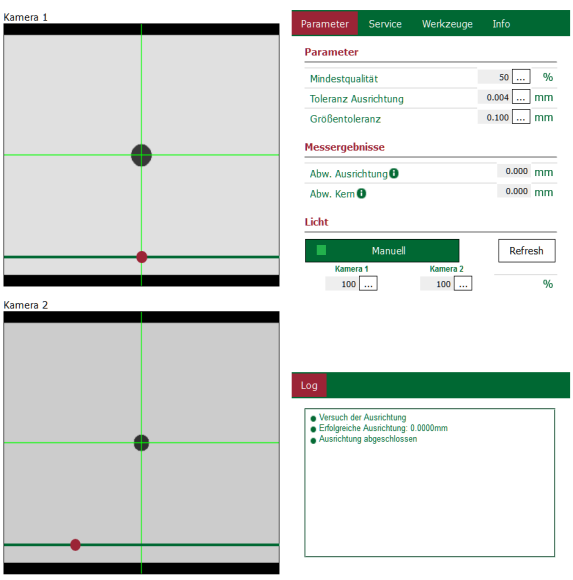

Abbildung 1.2: Interface nach dem Redesign

Weitere Modifikationen am Framework umfassten vor allem die Vereinfachung und Lesbarkeit des Codes. Die Hauptaufgabe in dieser Phase war das Verständnis, da dies die Grundlage für weitere Arbeit am Javascript ist.

1.2 Einschub: Filterfunktion für SVN-Repositories

Da mein Betreuer, welcher mit mir gemeinsam für die Entwicklung des neuen IVS-Systems verantwortlich war, krankheitsbedingt eineinhalb Wochen fehlte, war die weitere Arbeit an der Weboberfläche für diese Zeit leider nicht möglich. In der Zwischenzeit wurde mir dafür die Verbesserung der Oberfläche des Intranets aufgetragen.

Das Intranet enthält eine Liste von SVN-Repositories, die in Form eines Baumes dargestellt werden. Die einzelnen Einträge sind dabei in verschiedenen Ebenen enthalten. Im folgenden Codebeispiel (HTML) wird dies deutlich:

```
<ul class="mktree" id="mktree1">
  <li id="1">Firma 1</li>
    <ul>
      <li id="1-1">Repository 1-1</li>
      <li id="1-2">Repository 1-2</li>
    </ul>
  </li>
  <li id="2">Repository 2</li>
  <li id="3">Firma 3
    <ul>
      <li id="3-1">Repository 3-1</li>
      <li id="3-2">Projekt 3-2
        <ul>
          <li id="3-2-1">Repository 3-2-1</li>
```

```
15          <li id="3-2-2">Repository 3-2-2</li>
16        </ul>
17      </li>
18    </ul>
19  </li>
20 </ul>
```

Listing 1.1: Beispielcode zu den SVN-Repositories im Intranet

Zur Darstellung des Baumes in aus- und einklappbarer Form wurde das schon vorhandene Script `mktree` von Matt Kruse verwendet.

Meine Aufgabe bestand nun darin, eine Suchfunktion für Einträge in dieser Liste zu implementieren. Dazu habe ich mithilfe von `JavaScript` und `jQuery` gearbeitet und einige Funktionen geschrieben. Diese Funktionen basieren zudem auf dem oben genannten Script (`mktree`) und bauen auf dessen Funktionalität auf.

```
1  function filter() {
2    var input = document.getElementById('searchInput');
3    var filter = input.value.toLowerCase();
4    var url = window.location.href.split("?")[0];
5    document.getElementById("searchURL").href = url + "?filter=" + filter;
6    //Search through all list items
7    var li = $('li');
8    for (var i = 0; i < li.length; i++) {
9      //Get all parents of this node which have the filter.
10     var parents = $(li[i])
11       .parents('li')
12       .filter(function() {
13       return this.textContent
14         .trim()
15         .split('␣␣␣␣')[0]
16         .toLowerCase()
17         .indexOf(filter) > -1;
18     });
19     //Select the correct String to search in.
20     var str = li[i]
21       .textContent
22       .split('␣(')[0] //Remove the ( browse / ...)
23       .toLowerCase()
24       .trim()
25       .split("␣␣␣␣")[0]; //Bug fix
26     if (str.indexOf(filter) > -1 || parents.length) {
27       //Item or Parent contains filter
28       li[i].style.display = ""; //Display normally
29       $(li[i]).parents().each(function() {
30         this.style.display = "";
31       });
32       if (li[i].id != "") {
33         expandToItem('mktree1', li[i].id); //Expand tree
34       }
35       window.scrollTo(0, 0); //Fixes weird behaviour
36     } else {
37       li[i].style.display = "none"; //Hide item
```

```
38      }
39    }
40  }
```

Listing 1.2: Filterfunktion für das Intranet

Die oben geschriebene Funktion wird aufgerufen, sobald sich der Eingabewert des `<input>`-Felds (siehe Zeile 8) ändert. Eine Änderung der Eingabe hat somit zur Folge, dass die darunter stehende Liste von Repositories neu gefiltert wird und nicht passende Einträge entsprechend versteckt werden.

Weiterhin wird die Funktionalität von `mktree` genutzt, um den Baum zu den angezeigten Repositories automatisch aufzuklappen. Dies wird durch einen einfachen Aufruf der richtigen Funktion erreicht.

Nach der Implementierung dieser Funktionalität habe ich noch eine weitere Verbesserung an der Seite durchgeführt, wodurch sich der Cursor nicht im Textfeld befinden muss, um zu suchen. Dies hat den Vorteil, dass direkt nach dem Laden der Webseite oder auch nach dem manuellen Ein- bzw. Ausklappen des Baumes ohne weitere Mausbetätigung gesucht werden kann.

```
1  document.onkeypress = function(event) {
2    //Variables for search field and pressed key
3    var input = document.getElementById('searchInput');
4    var key = event.key;
5    if (document.activeElement !== input && key.length === 1) {
6      //Not focused => Put the pressed key in the search bar.
7      input.value += key;
8    }
9    //Jump to the search field.
10   input.focus();
11 }
```

Listing 1.3: Funktion für Suche auf Tastendruck

Um dies zu erreichen, habe ich eine Funktion geschrieben, welche beim Tastendruck aufgerufen wird. Die gedrückte Taste wird gespeichert und das Textfeld fokussiert. Handelt es sich bei der Taste um einen Buchstaben (nicht etwa `backspace`, daher `key.length === 1`), so wird der Wert der Taste zusätzlich ins Textfeld geschrieben, um einen doppelten Tastendruck zu vermeiden.

Eine letzte Optimierung der Seite erfolgte durch die Möglichkeit, eine Suchanfrage in Form der URL zu speichern (ein sogenannter Permalink) und somit weiterleitbar zu machen. Meine Vorgehensweise dafür war, den Filter mit dem entsprechenden Tag `?filter=` an die URL anzuhängen. Beim Laden der Seite wird die URL dann nach dem entsprechenden Tag durchsucht, und falls ein Tag gefunden wird, wird der Inhalt des Suchfeldes gesetzt und die Filterfunktion aufgerufen.

Weiterhin wird auf der Webseite ein Hyperlink dargestellt, welcher immer zur aktuellen Suchanfrage verlinkt. Durch Kopieren dieses Links kann die entsprechende Suche also weitergeleitet werden. Erreicht wird dies durch das Aktualisieren der dabei hinterlegten URL nach dem Aufruf von `filter()`.

```
1  document.ready = function() {
```

```
2   //First, let's do some ugly bug fixing for mktree (-_-)
3   setDefault("treeClass","mktree");
4   setDefault("nodeClosedClass","liClosed");
5   setDefault("nodeOpenClass","liOpen");
6   setDefault("nodeBulletClass","liBullet");
7   setDefault("nodeLinkClass","bullet");
8   setDefault("preProcessTrees",true);
9
10  var href = window.location.href;
11  //Filter the URL to find the filter.
12  var filter = undefined;
13  try {
14    filter =  href.split('?')[1] //after first "?"
15              .split('filter=')[1] //after "filter="
16              .split('&')[0]; //stop at next "&"
17  } catch (e) {}
18  if (filter != undefined) {
19    //Filter found => search for it
20    document.getElementById('searchInput').value = filter;
21    document.getElementById('searchInput').onkeyup();
22  }
23  document.getElementById("searchURL").href = window.location.href;
24 }
```

Listing 1.4: Filterung beim Laden der Seite

Beim Laden der Webseite wird nun also der Filter in der URL ausgelesen und ins Textfeld eingefügt.

1.3 Dokumentation des Frameworks

Eine weitere wichtige Aufgabe war das Dokumentieren des bereits vorhandenen Codes. Da der ursprüngliche Autor leider nicht mehr bei isys vision angestellt war und ich mich nun schon länger damit auseinander gesetzt hatte, sollte nun eine umfassende Dokumentation des Aufbaus erstellt werden.

Dies erleichtert einerseits die Arbeit am Framework selbst (Änderungen und Updates beispielsweise) und bietet denjenigen, die es nur verwenden (z.B. für eine neue Webseite mit demselben Framework) ein einfaches, gut verständliches Interface. Weiterhin können komplexere Vorgänge wie die Datenübertragung anschaulich erklärt und mithilfe von Diagrammen verdeutlicht werden.

1.3.1 Codekommentierung mit JSDoc

Im ersten Schritt arbeitete ich dabei nur am Quelltext selbst, den ich zuvor schon in lesbare Form gebracht hatte. Durch zusätzliche Kommentare an wichtigen und irreführenden Stellen konnte ich dabei eine erste Grundlage schaffen. Die größte Änderung war die Einführung einer Codedokumentation im JSDoc-Format, ähnlich zu den bei Java eingesetzten JavaDoc-Kommentaren. Die Vorteile davon liegen auf der Hand: Jede Methode bzw. Funktion hat somit eine eigene Beschreibung, in der Funktionsweise, Parameter, Rückgabewert usw. beschrieben werden.

Ein weiterer Vorteil von JSDoc ist die Möglichkeit, die Dokumentation automatisch zu generieren. Dies ist mit dem JSDoc-Tool von `Node.js` möglich, was nach der Installation einfach über den Befehl `jsdoc file.js` aufgerufen kann. Somit wird ein gut strukturiertes Dokument (inklusive Referenzen) automatisch erstellt.

1.3.2 Beschreibung der API

Nachdem der Quellcode selbst dokumentiert war, konnte ich mich nun mit den internen Abläufen des Frameworks beschäftigen und diese anhand von Grafiken anschaulich machen. In diesem Fall ging es vor allem um die Daten- und Befehlsübertragung vom IVS zum Webinterface und umgekehrt.

Die Übertragung der Daten erfolgt hierbei über eine `XMLHttpRequest`, bzw. (bei älteren Browsern) über ein `ActiveXObject`. Die Befehlscodes sind dabei im HTML-Code der betreffenden Buttons gespeichert:

```
1 <input type="text" size="6" maxlength="6" name="lms/SheetF.MinQuality">
2 <button name="lms/Cmd" value="10">Ausrichtung</button>
3 <button name="lms/Cmd" value="20">
4   <ul>
5     <li>LmsScb/Mot.RefOk</li>
6     <li>Referenzfahrt</li>
7   </ul>
8 </button>
```

Listing 1.5: Ins HTML eingebettete Befehle für das IVS

In Zeile 1 wird beispielsweise das Register `lms/SheetF.MinQuality` geschrieben und auf einen sechsstelligen Wert gesetzt, der vom Nutzer eingegeben wird.
In Zeile 2 befindet sich ein einfacher Befehlsbutton, der den Wert `10` ins Register `lms/Cmd` schreibt (beim IVS ist dies der Code für eine Ausrichtung).
In den Zeilen 3 - 8 findet sich nun ein Knopf mit Status, der

- den Wert `20` ins Register `lms/Cmd` schreibt und somit eine Referenzfahrt auslöst
- innerhalb der unsortierten Liste (Zeile 5) den Wert `LmsScb/Mot.RefOk` liest (Status der Referenzfahrt).

Die Liste (`<ul>`) dient dabei ausschließlich der Darstellung und hat keine Auswirkung auf die Funktion; die Listenelemente werden innerhalb des Buttons entsprechend formatiert. Weiterhin wird je nachdem, was im gelesenen Register `LmsScb/Mot.RefOk` steht, ein entsprechendes Label gesetzt (grün falls Referenzfahrt OK, rot falls nicht).

Im Javascript-Framework wird nun ein Event-Listener für alle Buttons und Eingabefelder erstellt, der dann die entsprechenden Befehle erkennt. Diese werden in Form einer URL in einer Liste gespeichert, welche dann mithilfe der bereits genannten `XMLHttpRequest` bzw. des `ActiveXObjects` ans IVS gesendet werden. Die Antwort des IVS wird dann entsprechend weiterverarbeitet und beispielsweise in Form eines Kamerabildes direkt ausgegeben.

2. Integration von Jira und Confluence

Das nächste größere Projekt war die Integration der Atlassian-Produkte *Jira* und *Confluence* in den Firmenalltag. Diese beiden Programme sind für das Production Management gedacht und haben das Ziel, einen durchgängigen, geplanten und leicht nachvollziehbaren Arbeitsfluss zu erreichen.

Confluence ist dabei ein einfaches System, dessen Ziel die einfache Weitergabe von Informationen ist. Es dient der Erstellung von Dokumenten, welche externe Daten wie beispielsweise Microsoft-Office-Dokumente oder YouTube-Videos sowie Bilder usw. enthalten können und vereinfacht die Zusammenarbeit mehrerer Personen an einem Dokument.

Jira arbeitet im Zusammenspiel mit Confluence und ist ursprünglich für die Softwareentwicklung gedacht. Entwicklungsvorgehensweisen wie *Scrum* o.Ä. können hier ohne Umwege umgesetzt werden und Jira bietet Funktionalitäten wie Sprints oder Burndown-Charts an, um den Arbeitsfluss zu sichern.

Die Verwendung von Jira und Confluence sollte das davor eingesetzte (und veraltete) Tool *easyPM* ersetzen, welches für die Planung von Aufgaben, Abläufen und Verwaltung von Zeitspalten eingesetzt wurde. Informationen wie E-Mails, Anrufprotokolle oder sonstige Daten waren hier jedoch dezentral, bzw. konnten nicht direkt in easyPM gespeichert werden. Durch das Zusammenspiel der beiden neuen Systeme sollte dieses Problem behoben und die Speicherung der Daten konsistent werden.

2.1 Einrichtung der Datenbank

Jira und Confluence sind beides Tools, die über einen Server laufen und dann über den Webbrowser der Endgeräte aufgerufen werden können. Dies sichert zum einen die Zentralität der Datenspeicherung und -verarbeitung und entlastet andererseits die Nutzergeräte stark. Weiterhin ist die Nutzung eines firmeninternen Servers (welcher außerdem für andere Zwecke wie das Intranet schon vorhanden war) sinnvoll, da somit die Sicherheit der gespeicherten Daten gewährleistet wird.

Für den Aufbau des Webservers wird dabei sowohl von Jira als auch von Confluence eine Datenbank verwendet. Das kann sowohl eine *MySQL*-, *Oracle*- oder *Microsoft-SQL*-Datenbank sein, oder aber auch die Open-Source-Alternative *PostgreSQL*. Wegen der eher schlechten Unterstützung von MySQL habe ich schlussendlich eine PostgreSQL-Datenbank eingesetzt.

PostgreSQL ist ein freies und quelloffenes Datenbankmanagementsystem. Es wird von Betriebssystemen wie Windows, Linux und anderen UNIX-ähnlichen Systemen unterstützt und wird bei den meisten Linux-Distributionen bereits mitgeliefert.

Um den Linux-Server von isys zu simulieren und die Datenbank entsprechend einzurichten, habe ich eine virtuelle Maschine mit einer Ubuntu-Installation verwendet. Nach der Einrichtung des Systems konnte ich mit der Konfiguration des Datenbankservers beginnen. Nach einiger Verwirrung und etwas Recherche stellte sich heraus, dass für das Erstellen einer Tabelle in PostgreSQL ein Linux-Nutzername angegeben werden muss. Dies war praktisch, da die Jira- und Confluence-Systemdienste sowieso einen eigenen Account erstellten, den ich somit direkt mitverwenden konnte. Ich hatte nun also drei Zugänge:

- `ubuntu`, Super-User-Account und Hauptaccount des Rechners,
- `confluence`, für die Datenbank und den Systemdienst des Confluence-Servers,
- `jira`, für die Datenbank und den Systemdienst des Jira-Servers.

Mit einigen einfachen SQL-Befehlen waren die notwendigen Datenbanken dann auch schnell erstellt:

```
1 CREATE DATABASE confluencedb WITH ENCODING 'UNICODE';
2 GRANT ALL PRIVILEGES ON DATABASE confluencedb TO confluence;
3
4 CREATE DATABASE jiradb WITH ENCODING 'UNICODE';
5 GRANT ALL PRIVILEGES ON DATABASE jiradb TO jira;
```

Listing 2.1: SQL-Befehle zur Erstellung der Datenbanken

Zur Anbindung an Jira bzw. Confluence musste nun nur noch bei der Einrichtung jeweils die entsprechenden Datenbankdaten angegeben werden. Da hier der Jira- bzw. Confluence-Server und die Datenbank auf dem selben Rechner laufen, ist der Host beispielsweise `127.0.0.1` bzw. `localhost`.

Die Installation von Jira und Confluence erfolgte dann einfach über die Ausführung eines Shell-Skriptes und die Eingabe einer Lizenz. Somit war die Einrichtung der Systeme vollständig und ich konnte einige Einstellungen vornehmen, z.B. Anpassungen im Design der Seiten, sodass die Oberfläche den Farben der Firma entsprach (Grün [#006833] und Rot [#9B2536]).

2.2 Einrichtung des Intranetservers

Nun, da Jira und Confluence online und konfiguriert waren, setzte ih mich mit der Einrichtung eines Servers auseinander, welcher im Intranet betrieben werden und die beiden Webtools hosten sollte.

Im ersten Schritt machte ich den dafür zur Verfügung gestellten Rechner nutzbar, indem ich eine frische Linux-Installation vornahm. Auf dieser Basis konnte ich nun mit `VirtualBox` eine virtuelle Maschine einrichten; auch darauf lief ein Linux-Betriebssystem. Die Nutzung einer virtuellen Maschine hatte hierbei mehrere Vorteile:

1. **Flexibilität**: Durch das Speichern in einem **Disk Image** ist der Ort des Servers sehr leicht austauschbar. Durch Verschieben des Images sind sämtliche Daten des Servers übernommen, was die Wartung stark vereinfacht.
2. **Sicherheit**: Da die Daten in einem einzigen Image liegen, welches auf der Festplatte des Wirtrechners gespeichert wird, ist die Datensicherung bzw. Erstellung von Backups be-

sonders komfortabel. Mithilfe eines RAID-Systems wird in diesem Fall die Datensicherheit gewährleistet und Backups erstellt.

3. **Vielseitigkeit**: Der Server läuft in einer virtuellen Maschine, welche auf die Ressourcen des Wirtrechners zugreift. Durch das Einrichten einer weiteren virtuellen Maschine kann dessen Rechenleistung und Speicherkapazität optimal ausgenutzt werden und die Maschine somit auch für andere Zwecke verwendet werden.

Nach erneuter Einrichtung der in Kapitel 2.1 beschriebenen Datenbank und der Anbindung an Jira/Confluence musste ich mich nun noch mit der Netzwerkverbindung auseinandersetzen. Die zugeordnete IP-Adresse sollte dabei `10.0.0.14` sein; Jira und Confluence jeweils unter den Standardports `8080` bzw. `8090` erreichbar sein.

Als Erstes musste die virtuelle Maschine über eine Netzwerkbrücke auf das lokale Netzwerk zugreifen können. Dies war schnell eingestellt; eine einfache Änderung im VirtualBox-Interface genügte dafür. Mithilfe der Netzwerkbrücke wird hier das lokale Netzwerk (isys vision Intranet) mit dem von der virtuellen Maschine erstellten Netzwerk verbunden, sodass diese die gehosteten Services auf die Webports anlegen kann.

Im nächsten Schritt braucht die virtuelle Maschine eine feste (statische) IP-Adresse. In diesem Fall sollte dies die `10.0.0.14` sein. Um diese zuzuweisen, kann unter Linux die Datei `/etc/network/interfaces` bearbeitet werden:

```
1 auto enp0s3 #Name der Netzwerkbrücke
2 iface enp0s3 inet static #IP auf Statisch ändern
3 #Setzen von Verbindungsinformationen
4 netmask 255.255.0.0
5 gateway 10.0.0.10
6 network 10.0.0.0
7 broadcast 10.0.0.255
8 dns-nameservers 10.0.0.10
```

Listing 2.2: Änderungen in `/etc/network/interfaces`