

# Software Projekt - Excel als SQL Datenbank

Authoren: Rijad Žužo, Séverin Müller

Dozent: Ulrich Hauser

"xl2DB"

Version: 0.9 vom 31.05.2015



## Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>1</b>
1.1	Ausgangssituation . . . . .	1
1.2	Lösungsidee . . . . .	1
<b>2</b>	<b>Anforderungsliste</b>	<b>2</b>
2.1	Muss-Anforderungen . . . . .	2
2.2	Soll-Anforderung . . . . .	2
2.3	Wunsch-Anforderung . . . . .	2
<b>3</b>	<b>Projektumgebung</b>	<b>3</b>
3.1	Entwicklungsprozess . . . . .	3
3.2	Ablauf . . . . .	3
3.3	Programmiersprache . . . . .	3
3.4	Verwaltungssystem . . . . .	3
<b>4</b>	<b>Projekt Planung</b>	<b>4</b>
4.1	Zeitplan . . . . .	4
4.2	Product Backlog . . . . .	4
4.3	Sprints . . . . .	5
4.4	Sprint Stand-up Meetings . . . . .	5
4.5	Sprint Review . . . . .	5
<b>5</b>	<b>Ablauf der Entwicklung</b>	<b>6</b>
5.1	Modellierung . . . . .	6
5.1.1	Klassendiagramm . . . . .	6
5.1.2	Zustandsdiagramm . . . . .	7
5.1.3	Sequenzdiagramm . . . . .	8
5.2	GUI . . . . .	9
5.2.1	Hauptfenster . . . . .	9
5.2.2	Aufrufbare Fenster . . . . .	10
5.3	Database Handling . . . . .	10
5.4	Backup Management . . . . .	10
5.5	Stunden Journal . . . . .	11
<b>6</b>	<b>Testen</b>	<b>12</b>
<b>7</b>	<b>Ziel</b>	<b>14</b>
7.1	Resultat . . . . .	14
7.2	Resultat: Muss-Anforderungen . . . . .	14
7.3	Resultat: Soll-Anforderung . . . . .	14
7.4	Resultat: Wunsch-Anforderung . . . . .	14
7.5	Gelerntes . . . . .	15
<b>8</b>	<b>Selbstständigkeitserklärung</b>	<b>15</b>

## Abbildungsverzeichnis

1	Zeitplanung . . . . .	4
2	Das Produkt Backlog . . . . .	4
3	Sprints 1 - 3 . . . . .	5
4	Sprints 4 - 5 . . . . .	5
5	Klassendiagramm . . . . .	6
6	Zustandsdiagramm . . . . .	7
7	Sequenzdiagramm . . . . .	8
8	Hauptfenster . . . . .	9
9	Aufrufbare Fenster . . . . .	10
10	Stunden Journal der Software Entwicklung . . . . .	11

# 1 Motivation

Das Projekt wurde von uns im Rahmen des Software Projekt Moduls für den Bosnischen Club St. Gallen erarbeitet. Diese führen seit langem eine Excel-Liste für die Mitgliederverwaltung. Für Sie war dies das einfachste Werkzeug, jedoch gab es immer wieder Probleme, wie zum Beispiel das ungewollte löschen ganzer Zeilen, oder das verrutschen in den Zeilen / Spalten. Wir wollten jedoch nicht eine komplett neue Datenbank erarbeiten, und entschieden uns, Ihnen ein Werkzeug für die Verwaltung der vorhandenen Excel Tabelle zur Verfügung zu stellen, welches eine einfache, intuitive GUI zur Verfügung stellt.

## 1.1 Ausgangssituation

Nebst den Problemen bei unaufmerksamer Bearbeitung, ist das Auslesen eines grösseren Excel Files deutlich langsamer als bei einer SQL Datenbank der gleichen bzw. einer vielfachen Grösse, deshalb ist es sinnvoll das Excel File in eine SQL Tabelle umzuwandeln und so zu Verarbeiten.

Die Konvertierung würde zusätzliche Software Kenntnisse erfordern, so griffen wir auf Vorhandene Office Werkzeuge bzw. vorhandene Excel Tabelle zurück.

Mit unserer xl2dB Applikation wurde das Problem gelöst. Die Umwandlung ist dank OleDb Connector unnötig und somit kann das Excel File bestehend bleiben.

## 1.2 Lösungsidee

Wir legten vor allem Wert auf ein simples User-Interface mit allen nötigen Funktionen. Die Software verarbeitet das Excel File im Hintergrund als SQL Tabelle und verbindet sich mit entsprechenden Treibern über ein File das via GUI (Dateibrowser) eingebunden werden kann. Das ermöglichen uns die Microsoft.Office.Core<sup>1</sup> und Microsoft.Office.Excel.Interop<sup>2</sup> Treiber.

Mit unserer Applikation hat der User einen begrenzten Einfluss auf das File und kann so weniger Schaden am File anrichten. Schaden können auch gleichzeitige Lese- / Speicherzugriffe auf ein Shared File auf einem Netzwerklaufwerk. In unserer Applikation dauert die Verbindung nur kurz, bis die Daten gelesen/geschrieben wurden und danach wird das File wieder freigegeben. Parallele Bearbeitung ist in dieser Version nicht implementiert.

Für zusätzliche Datenintegrität wird jeweils ein Backup des Files erstellt und kann nötigen Falls zurück gespielt werden. Dies ist nicht die 'ultimative Lösung', aber wir konnten so die Sicherheit bei Veränderungen am Excel File erhöhen und trotzdem ein 'für jeden lesbares' Dateiformat weiter verwenden; dass zum Beispiel auch mit einem USB Stick übertragen und auf heimischen Computern (weiter-)bearbeitet werden kann.

---

<sup>1</sup><https://msdn.microsoft.com/en-us/library/microsoft.office.core.aspx>

<sup>2</sup><https://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel%28v=office.15%29.aspx>

## 2 Anforderungsliste

Um die Bedürfnisse der Verwalter dieser Excel Liste bestmöglich abdecken zu können, haben wir uns mit Ihnen zusammengesetzt und die nachfolgenden Anforderungen definiert.

### 2.1 Muss-Anforderungen

Die Applikation muss:

- M1:** das vorhandene Excel File als Datenbasis verwenden.
- M2:** das Excel File lesen und beschreiben können.
- M3:** nach dem Lesen bzw. Schreiben die Verbindung trennen.
- M4:** bei Änderungen im File Backups erstellen.
- M5:** dem User das Handling des Excel Files abnehmen.

### 2.2 Soll-Anforderung

Die Applikation soll:

- S1:** leicht Bedienbar sein.
- S2:** ein intuitives, einfaches User-Interface haben.
- S3:** kompatibel mit Windows XP<sup>3</sup> und höher sein.
- S4:** kompatibel mit Excel 2003<sup>4</sup> und höher sein.

### 2.3 Wunsch-Anforderung

Die Applikation könnte:

- W1:** eine Benutzungsanleitung haben.
- W2:** mehrere Sprachen unterstützen.
- W3:** dem User Hilfe anbieten.
- W4:** einträge sortieren.
- W5:** auf Grund von Kriterien farbig hervorheben.

**Anmerkung:** Der Verein hatte noch einige Anforderungen an die Software, die rein vom Betriebssystem abhängig sind, deshalb wurden diese hier vernachlässigt.

---

<sup>3</sup><http://windows.microsoft.com/de-de/windows/home>

<sup>4</sup><http://www.office.com>

## 3 Projektumgebung

### 3.1 Entwicklungsprozess

Wir entschieden uns für den iterativen Scrum-Prozess<sup>5</sup>. Es erschien uns Ideal, da wir jede Woche Dienstag mindestens zwei Lektionen Zeit hatten. Die Sprint Dauer passten wir aufgrund des Stundenplanes an (siehe: Projekt Planung). Die Rollen teilten wir wie folgt auf:

**Scrum Master:** Rijad Žužo

**Management:** Séverin Müller

**Product Owner & Entwicklung** Rijad Žužo, Séverin Müller

**Customer** Verein

### 3.2 Ablauf

Die Stand-up Meetings erfolgten immer am Anfang der Lektionen am Dienstag bei einem Kaffee oder bei schönem Wetter kurz draussen. Wir benötigten jeweils kaum 10 Minuten, auch dank der guten Vorbereitung jeweils. Unter der Woche haben wir uns oft via Google Hangouts<sup>6</sup> abgeglichen oder Fragen direkt mit der Freigabe eines Bildschirms besprochen. Dank des Online-Tools Trello<sup>7</sup> konnten wir die Aufgabenpakete sogleich zuteilen und die Meetings waren dank unserer Vorbereitung sehr speditiv.

### 3.3 Programmiersprache

Da der Verein zuvor bereits Microsoft Excel verwendete und als Betriebssystem Microsoft Windows verwendete, lag es nahe, dass wir uns für die Windows Spezifische Programmiersprache C# entschieden. Dies in Kombination mit der IDE Visual Studio<sup>8</sup> ermöglichte uns eine Praxis nahe und angenehme Entwicklung sowohl der Funktionalitäten, wie und auch des GUI's.

### 3.4 Verwaltungssystem

Die Auswahl des Verwaltungssystem war sehr einfach, da wir uns im Vorhinein geeinigt haben die Software als OpenSource<sup>9</sup> entwickeln zu wollen. Somit haben wir uns für das OpenSource Git Projekt Verwaltungssystem<sup>10</sup> entschieden und unser Code auf der Website [www.GitHub.com](http://www.GitHub.com) gehostet.

Mit Hilfe dieser Collaboration Plattform war es uns möglich, gleichzeitig, in unabhängigen Codeteilen, zu arbeiten und den jeweiligen Stand zu synchronisieren

---

<sup>5</sup>[http://en.wikipedia.org/wiki/Scrum\\_software\\_development](http://en.wikipedia.org/wiki/Scrum_software_development)

<sup>6</sup><https://google.com/hangouts>

<sup>7</sup><http://www.trello.com/>

<sup>8</sup><https://www.visualstudio.com/>

<sup>9</sup>[http://de.wikipedia.org/wiki/Open\\_Source](http://de.wikipedia.org/wiki/Open_Source)

<sup>10</sup><http://git-scm.com/>

## 4 Projekt Planung

### 4.1 Zeitplan

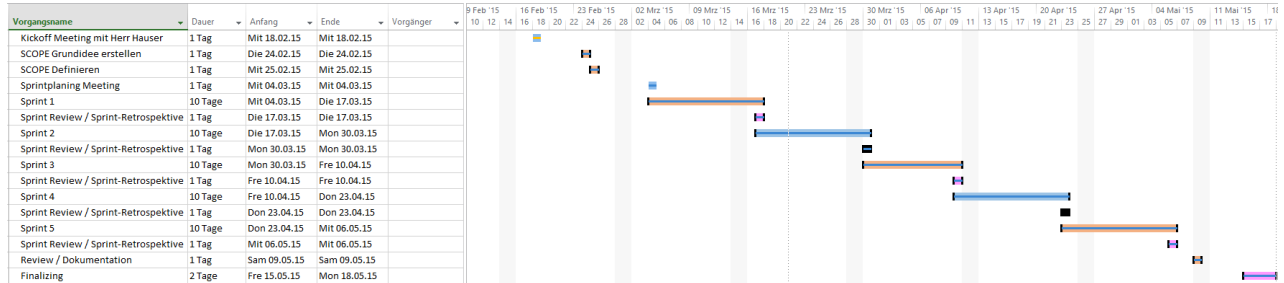


Bild 1: Zeitplanung

Wir haben uns auf eine Sprintdauer von 10 Tagen geeinigt. So hatten wir jeweils am Dienstag die Möglichkeit parallel weiter zu arbeiten und konnten Probleme oder offene Fragen klären. Die Sprints schlossen wir am Montag jeweils ab. Dies diskutierten wir entweder in der Waldau, St. Gallen am NTB oder kurz via Google Hangouts Videochat.

Das hat für uns sehr gut gepasst und die Atmosphäre sowie die Arbeiten waren angenehm. Die Sprintdauer war jeweils gerade richtig und der eingespielte Ablauf hat uns beiden sehr geholfen, wenn auch mal etwas dazwischen kam wie Prüfungen, Arbeit ... etc.

### 4.2 Product Backlog

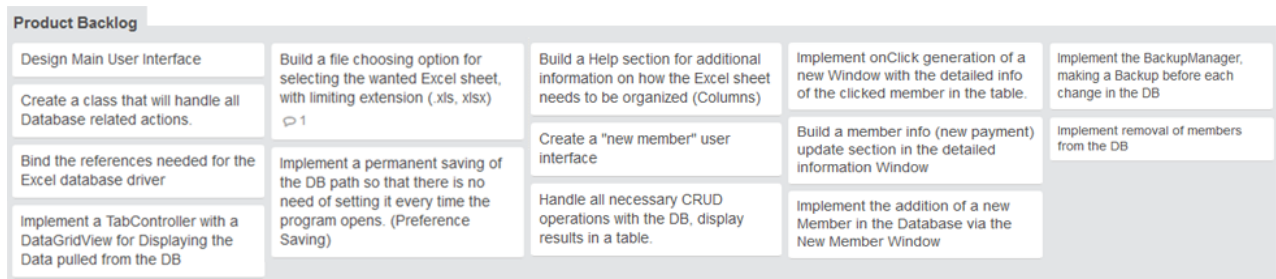


Bild 2: Das Produkt Backlog

Wie schon erwähnt haben wir für die Planung der Applikation die Website Trello.com benutzt. Das Produkt Backlog ist im Bild 2 abgebildet.

Das grösste Problem Anfangs war die Aufteilung dieses Backlogs in sinnvolle und erreichbare Sprints. Dies haben wir dynamisch gelöst.

Wir haben einige Features zur Implementation ausgewählt und in einen Sprint mit entsprechender Deadline gepackt. Konnte ein Feature nicht in der gegebenen Zeit implementiert werden, wurde der Task in den nächsten Sprint mitgenommen oder zurück gestellt.

So sind wir auf fünf Sprints gekommen die in Bild 3. und Bild 4. dargestellt sind.

## 4.3 Sprints

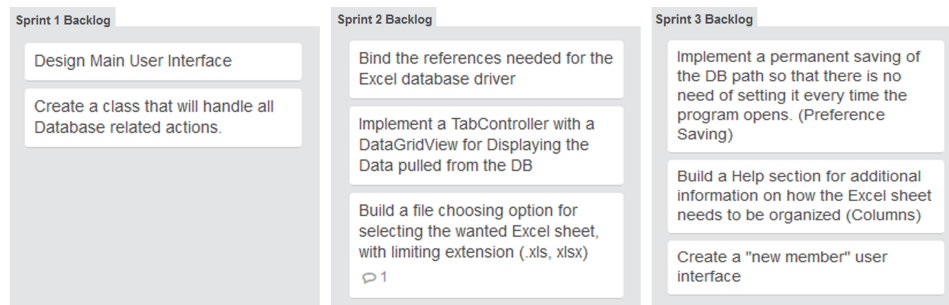


Bild 3: Sprints 1 - 3

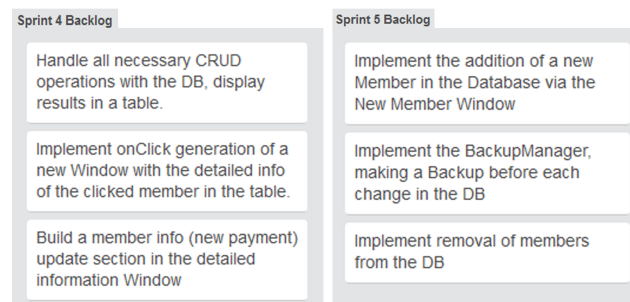


Bild 4: Sprints 4 - 5

## 4.4 Sprint Stand-up Meetings

Im Scrum Prozess werden die Meetings bewusst "Stand-up Meetings" genannt. Dies um unnötige Diskussionen beim bequemen Sitzen auf den Leder Stühlen eines in Glas gefassten Sitzungszimmers zu vermeiden.

Die Meetings sollen **kurz, prägnant und Informativ** sein. Dies haben wir auch immer versucht einzuhalten; kurz den Stand des jeweiligen Sprints und die darin befindlichen Tasks stehend besprochen und Probleme oder Ideen erst nachher am Laptop besprochen. Das hat im Grossen und Ganzen sehr gut funktioniert. Dies aber auch, weil wir immer wieder über das Projekt gesprochen, gechattet oder telefoniert hatten unter der Woche.

## 4.5 Sprint Review

Die Sprint Review fanden bei uns zumeist kurz nach den Stand-up Meetings statt, weil wir beides jeweils am Dienstag unter vier Augen besprochen haben. Hier hatten wir dank Trello auch eine sehr gute Übersicht und wussten anhand der Tasks immer was zu erledigen waren.

Wir denken aber, es wäre sinnvoll gewesen - vor allem bei einem grösseren Projekt - diese Reviews separat und in einem geschlossenen Team an einem abgemachten Termin zu besprechen. Um auch eventuelle Ideen noch besser einbringen zu können und noch mehr Brainstorming zu betreiben.

In unserem zweier Team aber hat das so gut funktioniert und es blieb so oder so nicht viel Zeit für noch mehr Ideen. Was aber nicht heissen soll, dass wir an dem Tool nicht noch das Eine oder Andere hinzufügen möchten.

## 5 Ablauf der Entwicklung

### 5.1 Modellierung

Die erste Phase der Entwicklung war das Strukturieren der Software - eine sehr wichtige Phase. Wir wollten ein klares Design von Beginn an, dass uns die Weiterentwicklung ermöglicht.

Hier haben wir sehr viel Zeit aufgewendet, mit den Beteiligten sehr viele Szenarien durchgespielt und uns selbst einige Tage Zeit gelassen um über den Funktionsumfang und die Gimmicks eine klare Vorstellung zu erhalten. Die GUI sollte möglichst einfach gehalten sein, was aber nicht heisst, dass sich wenige Funktionen dahinter verbergen.

#### 5.1.1 Klassendiagramm

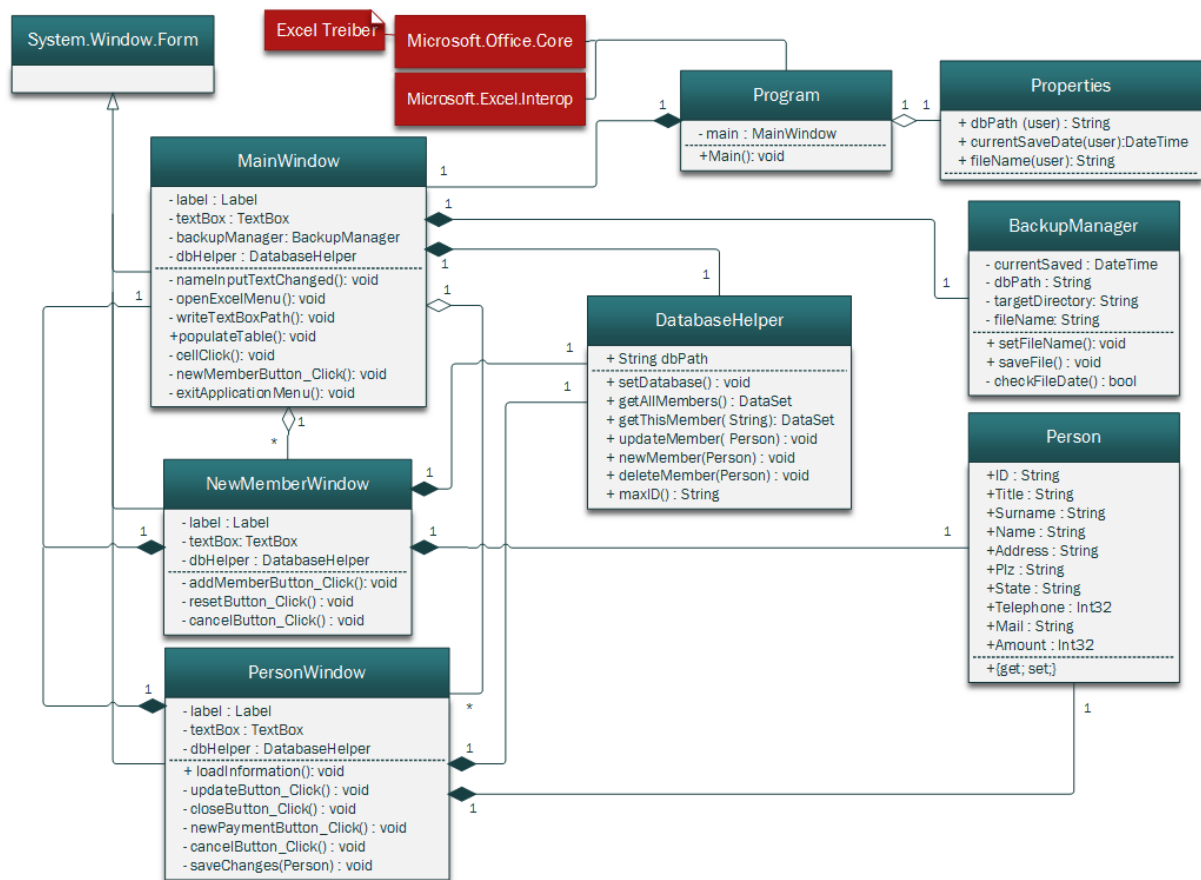


Bild 5: Klassendiagramm

Eintritt in die Applikation geschieht durch die Main() Methode in der Klasse Program. Diese ruft den Konstruktor von MainWindow auf welcher die weitere Logik übernimmt. Im Microsoft Visual Studio 2013 werden die GUI Gestaltung und Logik automatisch getrennt, deshalb wurde im Klassendiagramm nur der Logische-Teil dargestellt.

Die MainWindow Klasse ist der Core der Applikation. Dieser nimmt die Eingaben vom User entgegen und leitet sie den entsprechenden Methoden und Klassen weiter.

TODO//Die in Rot dargestellten Referenzen, die den OleDb Connector Beinhalten, ermöglichen uns die Verbindung zu Excel Files. Das Excel File dient als Datenbank und kann mit diesen Treibern Ausgelesen und Beschreiben werden. In der Klasse Properties sind die variablen die direkt im System des Users gespeichert werden beinhaltet, diese sind speziell von Visual Studio intern belegt und die sieht man im Code nicht.



### 5.1.2 Zustandsdiagramm

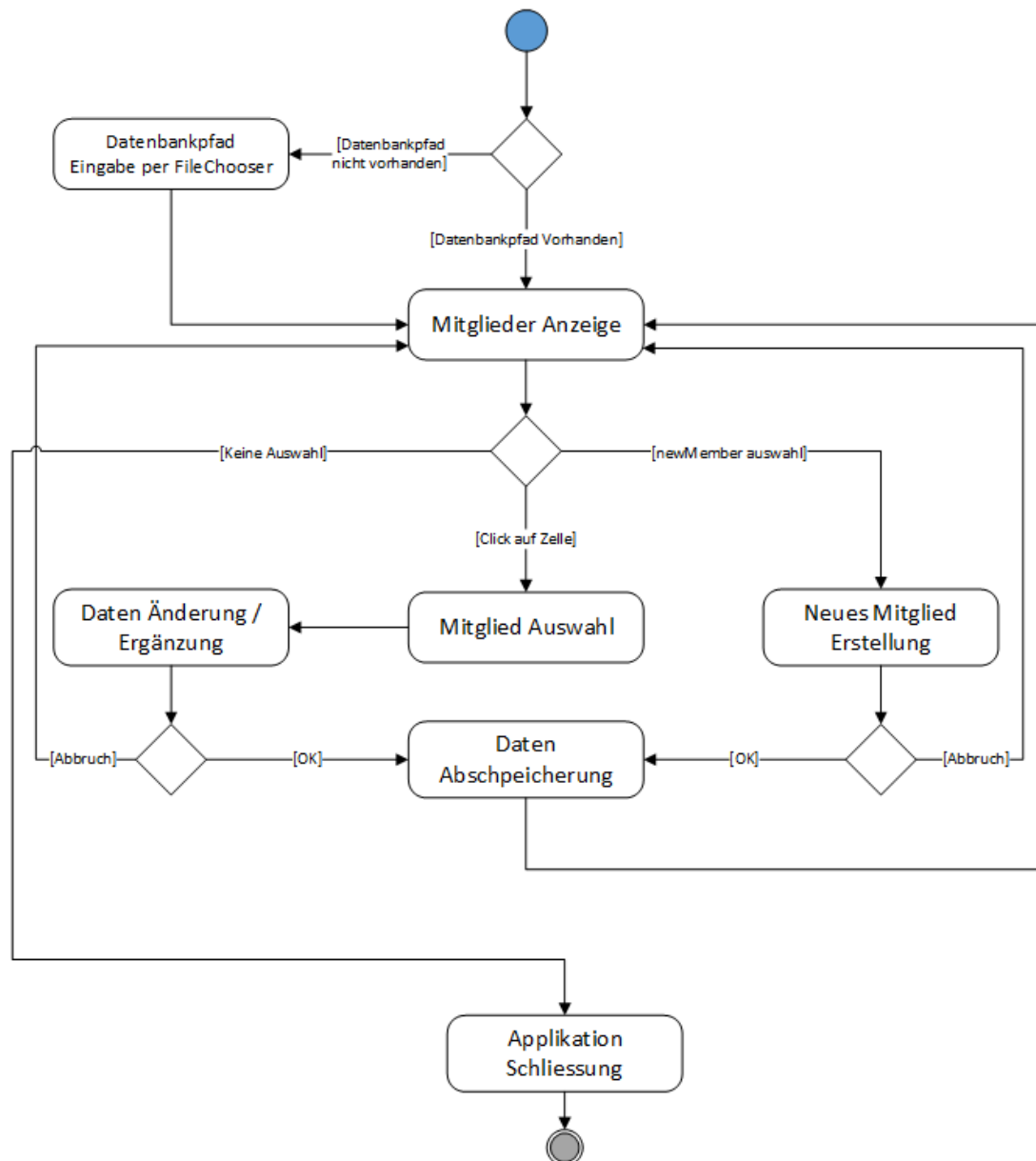


Bild 6: Zustandsdiagramm

Beim Start der Applikation wird überprüft ob bereits eine Datenbank hinterlegt ist oder nicht. Falls dieser vorhanden ist und verfügbar, werden alle Mitglieder automatisch aus der entsprechenden Datenbank ausgelesen und in der Tabelle dargestellt. Anderenfalls wird auf den User gewartet, bis dieser im File Chooser die gewünschte Datei (Datenbankpfad) manuell auswählt.

Nach dem Auslesen der Mitglieder kann der User weitere Aktionen ausführen, wie z.B. ein neues Mitglied anlegen oder Mitglieder auswählen und detaillierte Informationen zur Person ansehen oder ändern.

Bei jeder Aktion kann der User seine Änderungen speichern oder alles verwerfen. Die Datenbank wird jeweils aktualisiert.

### 5.1.3 Sequenzdiagramm

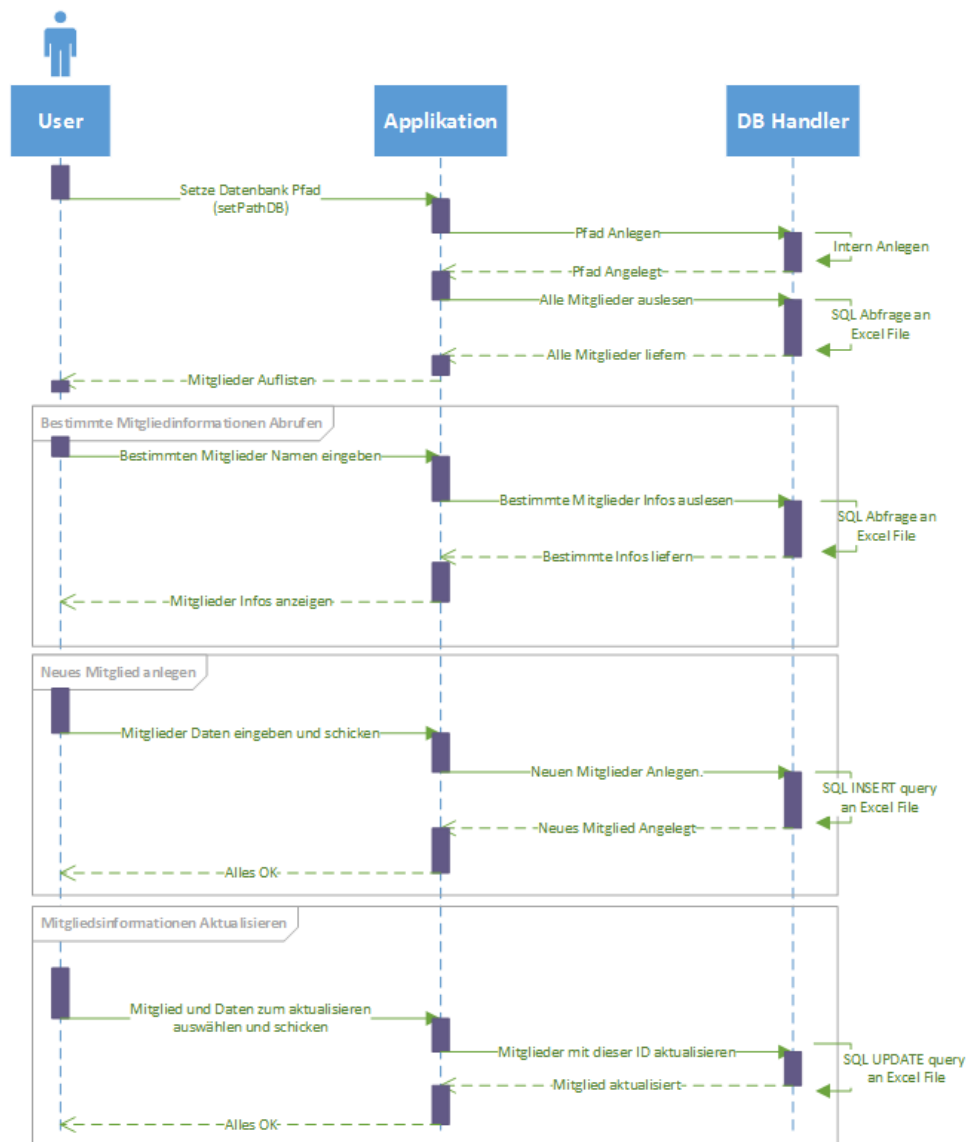


Bild 7: Sequenzdiagramm

Setzt der User einen Datenbankpfad werden automatisch alle Mitglieder ausgelesen und angezeigt. Im Sequenzdiagramm wurden mehrere Abläufe dargestellt:

#### Bestimmte Mitgliedinformationen Auslesen

Um nach einer Bestimmten Person zu suchen, gibt der User Name, Nachname, Adresse oder Ort ein. Jeder Tastenschlag setzt eine SQL Abfrage ab und entsprechende Datasets werden zurückgeliefert.

#### Neues Mitglied anlegen

Falls der User ein neues Mitglied anlegen will, öffnet er das entsprechende Formular, gibt die neuen Daten ein und drückt den Save Button. Alle Daten werden in ein Datamodel gepackt und in der Datenbank angelegt.

#### Mitgliedsinformationen Aktualisieren

Falls der User ein Mitglied in der Liste anklickt, wird das Mitglied angezeigt. Hier kann er die Informationen aktualisieren oder das Mitglied löschen. Beim Löschen wird einfach ein "NULL update" gesendet, da das direkte Löschen im File nicht vom Treiber unterstützt wird.

## 5.2 GUI

### 5.2.1 Hauptfenster

Die Oberfläche (GUI) wurde so einfach und intuitiv (selbsterklärend) wie möglich gestaltet. Dies auch auf Wunsche des Kunden:

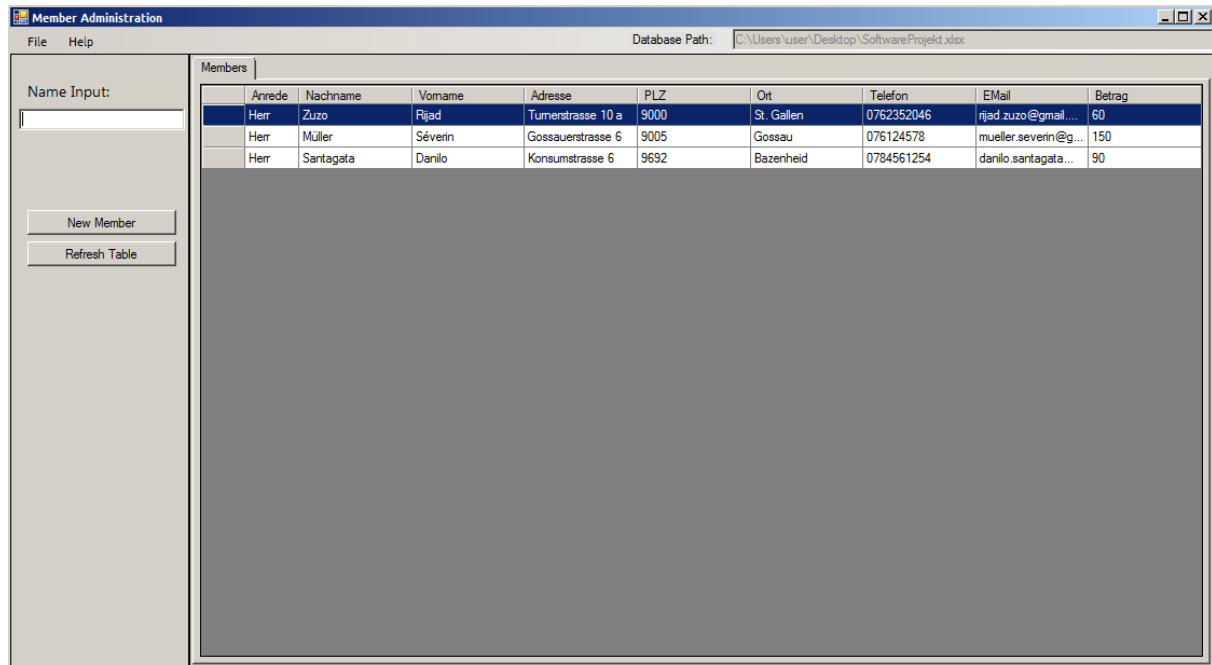
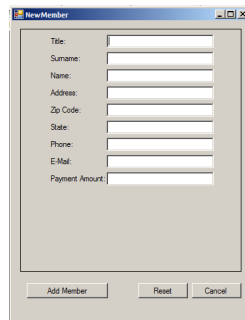


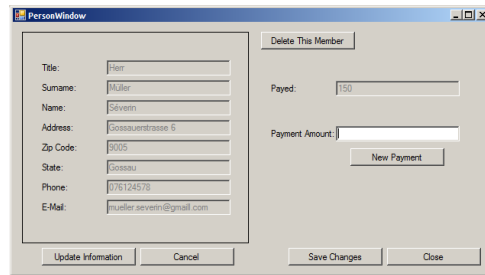
Bild 8: Hauptfenster

- Das Excel File kann via FileChooser eingebunden werden
- Oben rechts wird der Pfad zur Datei "Database Path" angezeigt.
- Ein einfaches Texteingabe (text-input) Feld "Name Input" links, ermöglicht "on text change" (Eingaben werden sofort übergeben) die Suche nach: Nachname, Vorname, Adresse, Ort.
- Ein Klick auf ein Mitglied in der Tabellenansicht öffnet ein neues Fenster (Bild 9.b) und zeigt alle erfassten Details dessen.
- Unter Help gibt es eine einfache Anleitung "Excel Sheet Format Example" und ein Info Dialog mit Kontaktinformationen "About".
- Der Button "New Member" öffnet das Fenster (Bild 9.a) zum erstellen eines neuen Mitgliedes mit den nötigen Informationen.
- Der Button "Refresh Table" lädt das Excel File erneut. (Nützlich bei unerwartetem applikations-behahmen)

### 5.2.2 Aufrufbare Fenster



(a) Neues Mitglied Formular



(b) Mitgliedsinformation Fenster

Bild 9: Aufrufbare Fenster

- Im Fenster "NewMember" (Bild 9.a), falls alle Felder im Formular ausgefüllt wurden, kann man die Schaltfläche "Add Member" betätigen welche den Speicherprozess ausführt.
- Beim klick auf "Reset" werden alle Felder zurückgesetzt und deren Text gelöscht. Ein klick auf "Cancel" schliesst das Formular.
- Im Fenster "PersonWindow" (Bild 9.b) kann durch die Schaltfläche "Update Information" das Mitglied bearbeitet werden oder mit der Schaltfläche "Delete This Member" gelöscht werden.
- Ausserdem ist es Möglich den einbezahlten Mitgliederbeitrag (Paid) zu erhöhen (positiver Betrag) oder ausstehende Beträge (negativer Betrag) ein zu pflegen (Payment Amount). Dies muss mit "New Payment" bestätigt werden.
- Ein Klick auf "Save Changes" speichert oder "Close" schliesst dieses Fenster ohne Speicherung.

**Anmerkung:** Alle Änderungen werden sofort in das Excel File geschrieben. Es gibt keine zusätzlichen Abfragen! Änderungen werden jedoch beim Klick auf Close im jeweiligen Fenster verworfen.

## 5.3 Database Handling

Wir haben die Klasse DatabaseHelper implementiert, in welcher alle Interaktion mit der Datenbank, in unserem Fall mit dem Excel File, stattfinden.

Aus den Referenzen Microsoft.Office.Core und Microsoft.Office.Excel.Interop haben wir den sogenannten OleDb Connector nutzen können. Dieser hat uns das Auslesen und Beschreiben des Excel Files ermöglicht, jedoch unterstützt es keine direkte Löschen Operation. Dies haben wir umgangen, in dem wir das Excel File mit NULL Werten Beschreiben und solche Zeilen im GUI (in der Tabelle) ignorieren.

Um Zugriffskollisionen zu vermeiden, wird nach jeder Interaktion mit dem Excel File die Verbindung unterbrochen. Der Verbindungsmanager hält die Verbindung aufrecht, solange mit dem File gearbeitet wird (lesen/schreiben). Das parallele bzw. gleichzeitige Bearbeiten des Excel Files ist in dieser Version noch nicht vollständig implementiert.

## 5.4 Backup Management

Wir unterstützen ein rudimentäres Backup des eingelesenen Excel Files. Die Datei wird bei der ersten Verwendung unter dem Pfad: %username%\Desktop\xlBackup gespeichert und umfasst jeweils eine Version beziehungsweise wird beim nächsten Aufruf überschrieben.

*Anmerkung:* Diese kann durch den Windows Schattenkopie Dienst (VSS) in weiteren Versionen geschützt werden.

## 5.5 Stunden Journal

Item Type	Sprint Number	Task Name	Time [h]
Sprint	1	Design Main User Interface	3.00
Sprint	1	Create a class that will handle all Database related actions.	3.00
Sprint	2	Bind the references needed for the Excel database driver	2.50
Sprint	2	Implement a DataGridView for Displaying the Data pulled from the DB	3.50
Sprint	2	Build a file choosing option for selecting the wanted Excel sheet, with limiting extension (.xls, .xlsx)	1.50
Sprint	3	Implement a permanent saving of the DB path so that there is no need of setting it every time the program opens. (Preference Saving)	2.50
Sprint	3	Build a Help section for additional information on how the Excel sheet needs to be organized (Columns)	2.50
Sprint	3	Create a new member user interface	2.50
Sprint	4	Handle all necessary CRUD operations with the DB, display results in a table.	9.00
Sprint	4	Implement onClick generation of a new Window with the detailed info of the clicked member in the table.	3.00
Sprint	4	Build a member info (new payment) update section in the detailed information Window	2.50
Sprint	5	Implement the addition of a new Member in the Database via the New Member Window	3.50
Sprint	5	Implement the BackupManager, making a Backup before each change in the DB.	4.00
Sprint	5	Implement removal of members form the DB	4.50
Software development total time			47.50

Bild 10: Stunden Journal der Software Entwicklung

Wir benötigten für das Projekt 47.50 Stunden. Im Anbetracht der Tatsache, dass wir in einer neuen Sprache und einem unbekannten IDE gearbeitet haben, sind wir zufrieden. Es war etwas mehr Aufwand als wir zuerst annahmen, jedoch hat uns das persönlich weitergebracht.

Für die Dokumentation in LaTeX<sup>11</sup> haben wir auch nochmals sehr viel Zeit aufgewendet. Freuten uns aber über die Fortschritte und sind am Schluss stolz, eine saubere, professionelle und ansprechende Dokumentation zu liefern.

Das Projekt ging ohne grössere Zwischenfälle - mit Ausnahme des Löschens - über die Bühne und die Software ist mehr als zufriedenstellend. Dies war unser erstes gemeinsames Projekt und wir sind uns einig, ein gutes Team zu sein!

<sup>11</sup><http://www.latex-project.org/>

## 6 Testen

### 1. GUI Tests (getestet am 10. Mai, 2015)

<b>1.1 Filechooser Aufruf:</b> Durchführung: Erwartetes Ergebnis:	Öffnen eines Filechooser Fensters. Im Menü: File -> Choose Excel File auswählen. Ein File Chooser Menu wird aufgerufen. Resultat: <b>Pass</b>	✓
<b>1.2 Excel File Auswahl:</b> Durchführung: Erwartetes Ergebnis:	Auswahl und Festsetzung eines Excel Files als Datenbank. Im File Chooser die gewünschte Datei auswählen und auf OK drücken. FileChooser wird geschlossen und die Tabelle mit Daten gefüllt. Resultat: <b>Pass</b>	✓
<b>1.3 Mitglied Infofenster:</b> Durchführung: Erwartetes Ergebnis:	Aufrufen des Informations-Fensters beim anklicken eines Mitglieds. In der Tabelle das entsprechende Mitglied anklicken. PersonWindow wird geöffnet mit allen Informationen über das Mitglied. Resultat: <b>Pass</b>	✓
<b>1.4 Neues Mitglied:</b> Durchführung: Erwartetes Ergebnis:	Aufrufen des "NewMember" Formulars. Klick auf die "New Member" Schaltfläche im Hauptfenster. NewMember Fenster wird aufgerufen. Resultat: <b>Pass</b>	✓
<b>1.5 Excel Beispiel:</b> Durchführung: Erwartetes Ergebnis:	Öffnen des Excel Format Beispiel Fensters. Im Menü: Help -> Excel Format Example auswählen. Fenster mit dem Excel Beispiel wird geöffnet. Resultat: <b>Pass</b>	✓
<b>1.6 Tabelle Refreshen:</b> Durchführung: Erwartetes Ergebnis:	Aktualisierung der Tabelle für die Anzeige neuer Daten. Klick auf die "Refresh Table" Schaltfläche im Hauptfenster. Die ganze Tabelle im Hauptfenster wird aktualisiert. Resultat: <b>Pass</b>	✓

### TODO// 2. Properties Speicherung Tests (getestet am 16. Mai, 2015)

<b>2.1 Pfad Speichern:</b> Durchführung: Erwartetes Ergebnis:	Der eingetragener Pfad im Filechooser wird im System des Users abgespeichert und wird beim nächsten Aufruf der applikation automatisch geladen. Datenbankpfad setzen und die Applikation neu starten. Datenbankpfad wird beim nächsten Applikationsaufruf in der GUI angezeigt. Resultat: <b>Pass</b>	✓
<b>2.2 Datum Speichern</b> Durchführung: Erwartetes Ergebnis:	Bei Aktulitätsüberprüfung, das neue Datum im System des Users abspeichern und bei nächster Überprüfung auslesen. Automatisch beim Backup Aktualitätscheck (ohne Änderung im File). Keine neue Kopie des Files erstellt (das alte Datum ist vorhanden). Resultat: <b>Pass</b>	✓

### 3. Datenbank Tests (getestet am 12. Mai, 2015)

<b>3.1 Auslesen aller Daten:</b> Durchführung: Erwartetes Ergebnis:	Bei bestehendem Datenbankpfad alle werte aus der DB auslesen. Automatisch, sofort nach Setzung des Pfades. Ein DataSet mit allen Daten wird zurückgegeben. Resultat: <b>Pass</b>	✓
<b>3.2 Auslesen einiger Daten:</b> Durchführung: Erwartetes Ergebnis:	Auslesen der Daten die mit der Eingabe im GUI. übereinstimmen. Text Eingabe im Eingabefeld unter ("Name Input:") in der GUI. Ein DataSet mit bestimmten Daten wird zurückgegeben. Resultat: <b>Pass</b>	✓
<b>3.3 Eingabe in DB:</b> Durchführung: Erwartetes Ergebnis:	Eingabe neuer Mitglieder in die Datenbank. Im "NewMember" Formular auf "Add Member" klicken. Ein neuer Eintrag entsteht in der Datenbank. Resultat: <b>Pass</b>	✓
<b>3.4 Bearbeiten der DB:</b> Durchführung: Erwartetes Ergebnis:	Alte Daten in der Datenbank mit neuen überschreiben. Im "PersonWindow" auf "Save Changes" klicken. Neue Daten sind in der Datenbank vorhanden. Resultat: <b>Pass</b>	✓
<b>3.5 Aktualisieren von Daten:</b> Durchführung: Erwartetes Ergebnis:	Mitgliederbeitrag erhöhen und verringern Betrag in "Payment Amount" eingeben und "New Payment" klicken. Betrag aktualisiert sich pos. bzw. neg. gemäss Eingabe im Feld Resultat: <b>Pass</b>	✓
<b>3.6 Löschen von Daten:</b> Durchführung: Erwartetes Ergebnis:	Ganze Zeilen (Mitglieder) in der Datenbank löschen. Im "PersonWindow" auf "Delete This Member" klicken. Gelöschtes Mitglied nicht mehr vorhanden in der Datenbank. Resultat: <b>Pass*</b>	✓

\* Da unser OleDb Connector mit dem ISAM Treiber das direkte löschen von Zeilen (SQL-DELETE) in einer gelinkten Liste nicht unterstützt, mussten wir dies wie vorhergehend erklärt umgehen. Jedoch funktioniert unsere Lösung nicht ganz wie erwartet. Der Bug wird erst offensichtlich wenn wir den letzten Eintrag in der Tabelle "löschen". Nach der "DELETE Erfolgsmeldung" kann nicht mehr mit der Datenbank interagiert werden. In diesem Fall liefert die Applikation auch im Debug Modus keine Fehlermeldungen, weshalb wir vermuten das dies ein Excel-Ordnungsproblem ist.

### 4. Backup Manager Tests (getestet am 13. Mai, 2015)

<b>4.1 Backup Erstellen:</b> Durchführung: Erwartetes Ergebnis:	Kopie des aktuellen Files im Pfad: %username%\Desktop\xlBackup platzieren. Automatisch, beim Start der Applikation. Ordner xlBackup auf Desktop mit Excel File vorhanden. Resultat: <b>Pass</b>	✓
<b>4.2 Backup Aktuell:</b> Durchführung: Erwartetes Ergebnis:	Falls die Backupdatei nicht aktuell ist, eine neue Kopie erstellen. Automatisch beim start der Applikation. Folder xlBackup im Desktop mit neuem (aktuellen) Excel File vorhanden. Resultat: <b>Pass</b>	✓

## 7 Ziel

### 7.1 Resultat

Die Software "xl2DB" erfüllt die Muss- & Soll-Anforderungen und erleichtert dem Verein das Handling der Mitglieder Datenbank nach ersten Rückmeldungen enorm. Dies ist für uns ein Erfolg.

Es sind noch Verbesserungen möglich, welche wir in Zukunft gerne noch einpflegen. Ausserdem sind die Wunsch-Anforderungen noch offen und bieten die Möglichkeit die Software noch attraktiver zu machen.

Wir haben in überschaubarer Zeit, erfolgreich eine Software entwickelt, die auch wirklich im Alltag eingesetzt werden kann. Der "Kunde" ist zufrieden und hat ein 'Wunsch' Werkzeug für sein Problem erhalten, dass ihn bei einem Individual-Softwarelieferanten ein ganzes Stück Geld gekostet hätte.

### 7.2 Resultat: Muss-Anforderungen

Die Applikation muss:

- M1:** das vorhandene Excel File als Datenbasis verwenden. ✓
- M2:** das Excel File lesen und beschreiben können. ✓
- M3:** nach dem Lesen bzw. Schreiben die Verbindung trennen. ✓
- M4:** bei Änderungen im File Backups erstellen. ✓
- M5:** dem User das Handling des Excel Files abnehmen. ✓

*Es sind alle Muss-Anforderungen erfüllt.*

### 7.3 Resultat: Soll-Anforderung

Die Applikation soll:

- S1:** leicht Bedienbar sein. ✓
- S2:** ein intuitives, einfaches User-Interface haben. ✓
- S3:** kompatibel mit Windows XP und höher sein. ✓
- S4:** kompatibel mit Excel 2003 und höher sein. ✓

*Es sind alle Soll-Anforderungen erfüllt.*

### 7.4 Resultat: Wunsch-Anforderung

Die Applikation könnte:

- W1:** eine Benutzungsanleitung haben. ✓ [rudimentär]
- W2:** mehrere Sprachen unterstützen. ✗
- W3:** dem User Hilfe anbieten. ✗
- W4:** einträge sortieren. ✗
- W5:** auf Grund von Kriterien farbig hervorheben. ✗

*Leider sind wir nicht dazu gekommen alle Wunsch-Anforderungen zu erfüllen. Hier gibt es Raum die Software weiter auszubauen und zu verbessern. Ein Projekt für die Zukunft!*



## 7.5 Gelerntes

In diesem Softwareprojekt haben wir sowohl den Software-Entwicklungs-Prozess "Scrum" angewendet, sowie eine neue Sprache C# kennengelernt.

"Scrum" wurde uns im ersten Teil von IuK näher gebracht. Jedoch ist es etwas anderes sich an die Prozesse und Abläufe halten zu müssen, als in der Theorie / Vorlesungsraum dem Dozenten zu hören zu müssen. Wir denken den Prozess erfolgreich umgesetzt zu haben, auch wenn es ein paar Stolper-Steine gab, und dies können wir in unseren Rucksack packen.

Die Sprache C# in Zusammenhang mit der Visual Studio Entwicklungsplattform war eine spannende Erfahrung. Besonders der Umgang mit den referenzierten Treibern und deren Eigenheiten, war für uns neu und benötigte einiges an Recherche und Testing. Jedoch war es unserer Meinung nach die Richtige Entscheidung auf diese Plattform zu setzten, da die Software auch ausschliesslich auf und für Microsoft Produkte entwickelt wurde.

## 8 Selbstständigkeitserklärung

Wir bestätigen hiermit, dass wir die vorstehende Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäss verwendete Textteile, Grafiken oder Bilder kenntlich gemacht haben. Diese Arbeit ist in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt worden.

Rijad Žužo

Séverin Müller

[Platzhalter digital]

---

[Platzhalter digital]

---