

Software Projekt - Excel als SQL Datenbank

Autoren: Rijad Žužo, Séverin Müller

Dozent: Ulrich Hauser



"xl2DB"

Version: 0.9 vom 24.05.2015

Inhaltsverzeichnis

1	Motivation	1
1.1	Ausgangssituation	1
1.2	Lösungsidee	1
2	Anforderungsliste	2
2.1	Muss-Anforderungen	2
2.2	Soll-Anforderung	2
2.3	Wunsch-Anforderung	2
3	Projektumgebung	3
3.1	Entwicklungsprozess	3
3.2	Ablauf	3
3.3	Programmiersprache	3
3.4	Verwaltungssystem	3
4	Projekt Planung	4
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
5	Ablauf der Entwicklung	6
5.1	Modellierung	6
5.1.1	Klassendiagramm	6
5.1.2	Aktivitätsdiagramm	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
6	Testen	11
7	Ziel	12
7.1	Resultat	12
7.2	Resultat: Muss-Anforderungen	12
7.3	Resultat: Soll-Anforderung	12
7.4	Resultat: Wunsch-Anforderung	12
7.5	Gelerntes	13
8	Selbstständigkeitserklärung	13

Abbildungsverzeichnis

1	Zeitplanung	4
2	Der Produkt Backlog	4
3	Sprints 1 - 3	5
4	Sprints 4 - 5	5
5	Klassendiagramm	6
6	Aktivitätsdiagramm	7
7	Sequenzdiagramm	8
8	Hauptfenster	9
9	Aufrufbare Fenster	10

1 Motivation

Das Projekt wurde von uns in der Freizeit 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 Konversion würde zusätzliche Software Kenntnisse erfordern, so griffen wir auf Vorhandene Office Werkzeuge zurück.

Mit unserer x2dB Applikation wurde das Problem gelöst. Die Umwandlung ist dank ODBC 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 und Microsoft.Office.Excel.Interop 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.

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.

2 Anforderungsliste

Um die Bedürfnisse der Verwalter dieser Excel Liste bestmöglich abdecken zu können, haben wir uns mit Ihnen zusammen gesetzt 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 und höher sein.
- S4:** Kompatibel mit Excel 2003 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 aber nichts mit der Applikation zu tun hatte sondern mehr vom OS abhängt, deshalb wurden diese hier vernachlässigt.*

3 Projektumgebung

3.1 Entwicklungsprozess

Wir entschieden uns für den iterativen Scrum-Prozess. Es erschien uns Ideal, da wir jede Woche Dienstag mindestens zwei Lektionen Zeit hatten. Somit entschieden wir uns, dass die Sprints 7 Tage dauerten und nicht wie gewöhnlich 30. Wir teilten die Rollen wie folgt auf:

Scrum Master: Rijad Zuzo

Management: Séverin Müller

Product Owner & Entwicklung Rijad Zuzo, 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. Unter der Woche haben wir uns oft via Google Hangouts abgeglichen oder Fragen direkt mit der Freigabe eines Bildschirms besprochen. Dank des Online-Tools Trello.com 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 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 entwickeln zu wollen. Somit haben wir uns für das OpenSource Git Projekt Verwaltungssystem entschieden und unser Code auf der Website www.GitHub.com gehostet.

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

4 Projekt Planung

4.1 Zeitplan

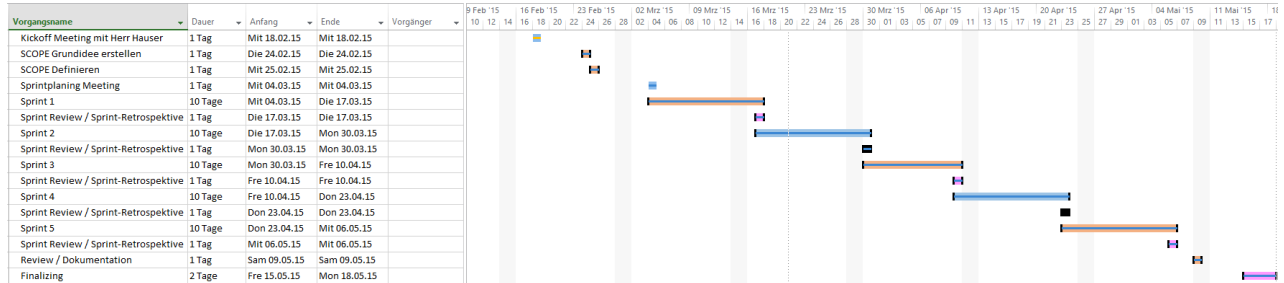


Bild 1: Zeitplanung

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

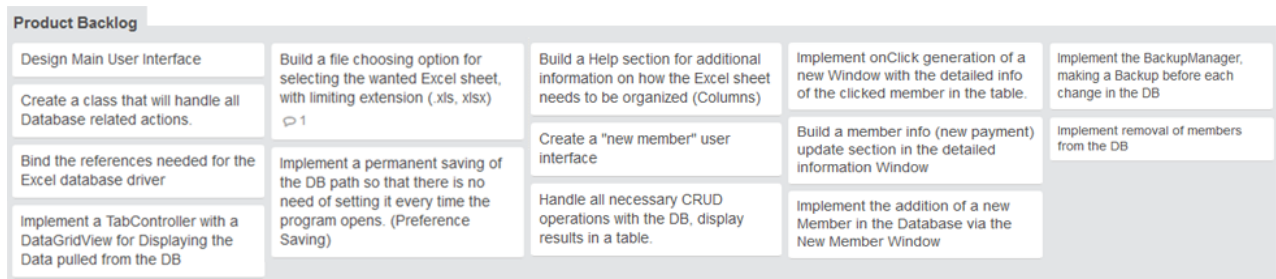


Bild 2: Der Produkt Backlog

Wie schon erwähnt haben wir für die Planung der Applikation die Website Trello.com benutzt wo wir das Produkt Backlog im Bild 2. definiert haben.

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

Wir haben einige Features zur Implementation ausgewählt und in einem Sprint mit einer entsprechenden Deadline eingepackt. Falls jeweilige Features nicht in der gegebenen Zeit implementiert werden konnten, wurden die in den nächsten Sprint weitergeleitet.

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

4.3 Sprints

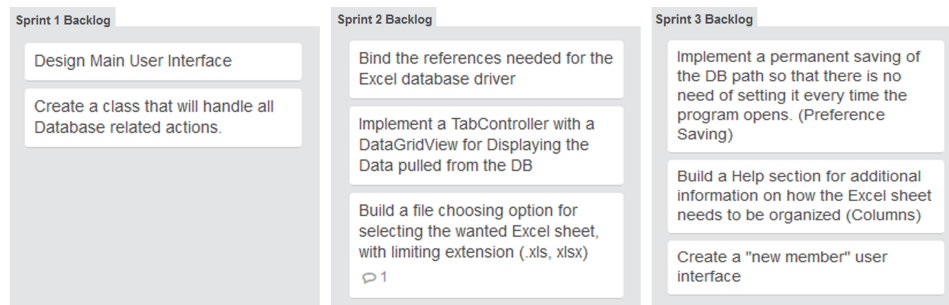


Bild 3: Sprints 1 - 3

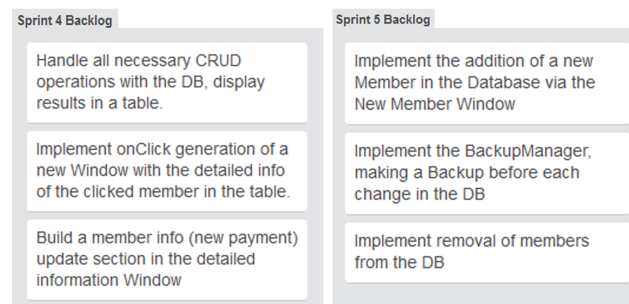


Bild 4: Sprints 4 - 5

4.4 Sprint Stand-up Meetings

Protokoll-artig auflisten was wir so besprochen haben usw.

4.5 Sprint Review

Review über all die Funktionalitäten und ob wir alles so erledigt haben wie es uns passt.

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

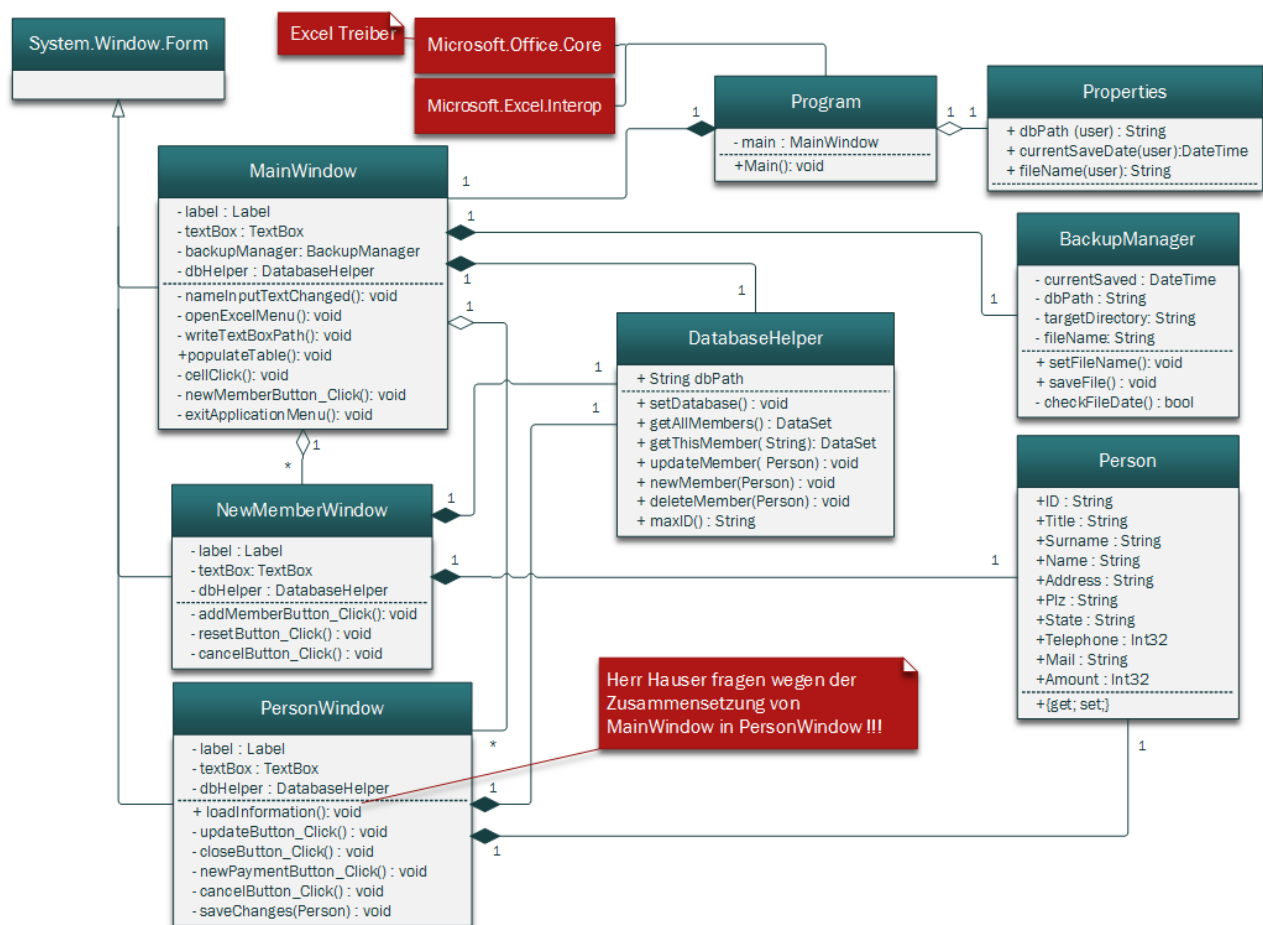


Bild 5: Klassendiagramm

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.

Die in Rot dargestellten Referenzen die den OleDb Treiber Beinhalten ermöglichen uns die Verbindung zu Excel Files. Das Excel File dient als Datenbank und kann mit diesen Treibern Asugelesen und Beschreiben werden.

5.1.2 Aktivitätsdiagramm

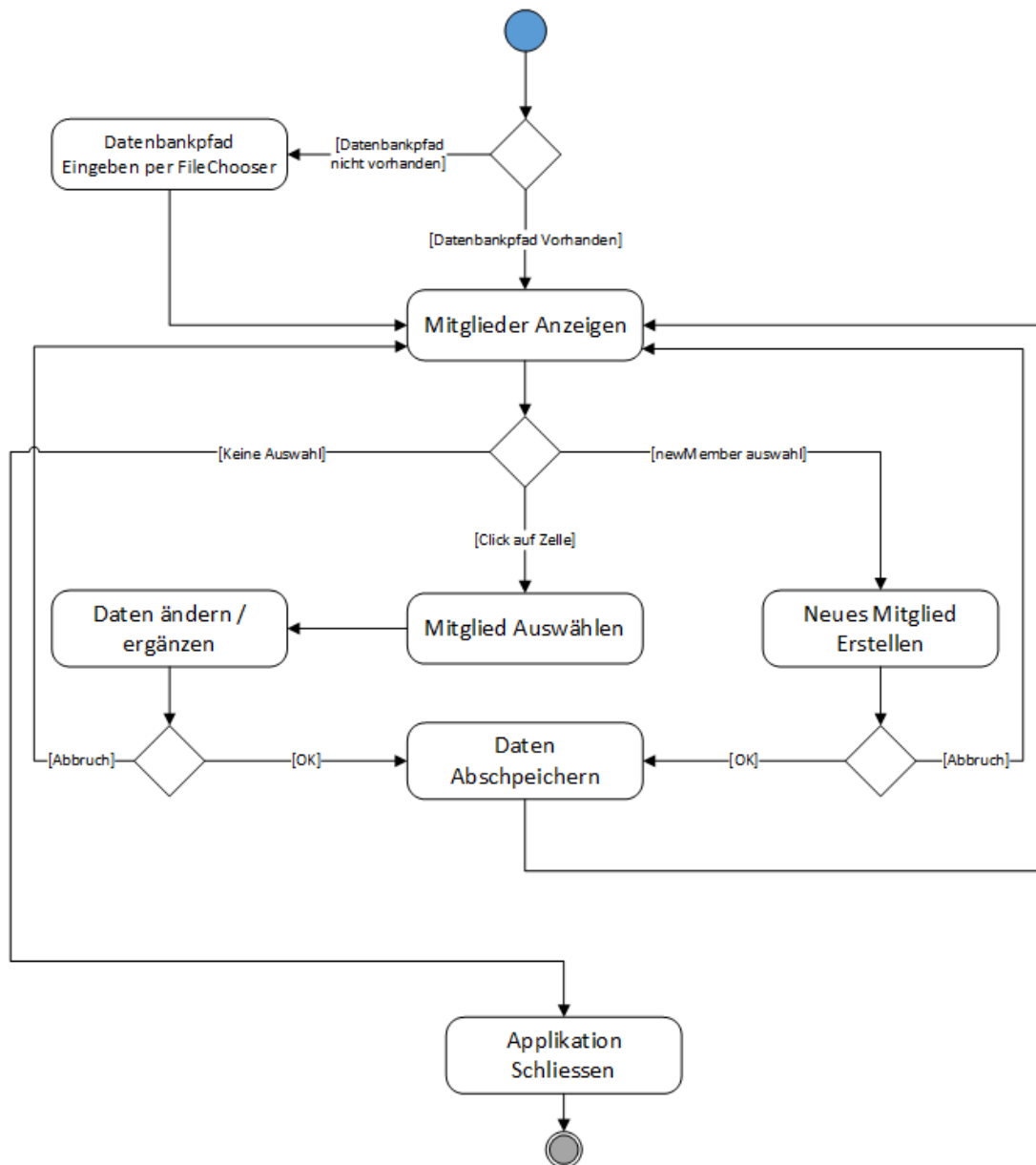


Bild 6: Aktivitätsdiagramm

Beim aufrufen der Applikation wird überprüft ob es einen Datenbankpfad schon gibt oder nicht. Falls es einen gibt werden alle Mitglieder automatisch aus der entsprechenden Datenbank ausgelesen und in der Tabelle dargestellt. Im anderen Fall wird auf den User gewartet bis er im File Chooser die gewünschte Datei (Datenbankpfad) auswählt.

Nach auslesen der Mitglieder kann der User weitere Aktionen ausführen wie, einen neuen Mitglied anlegen oder Mitglied auswählen und detaillierte Informationen ansehen oder ändern.

Bei jeder Aktion kann der User seine Änderungen Abspeichern oder alles verwerfen. Nach diesen Aktionen werden die Mitglieder nochmals ausgelesen.

5.1.3 Sequenzdiagramm

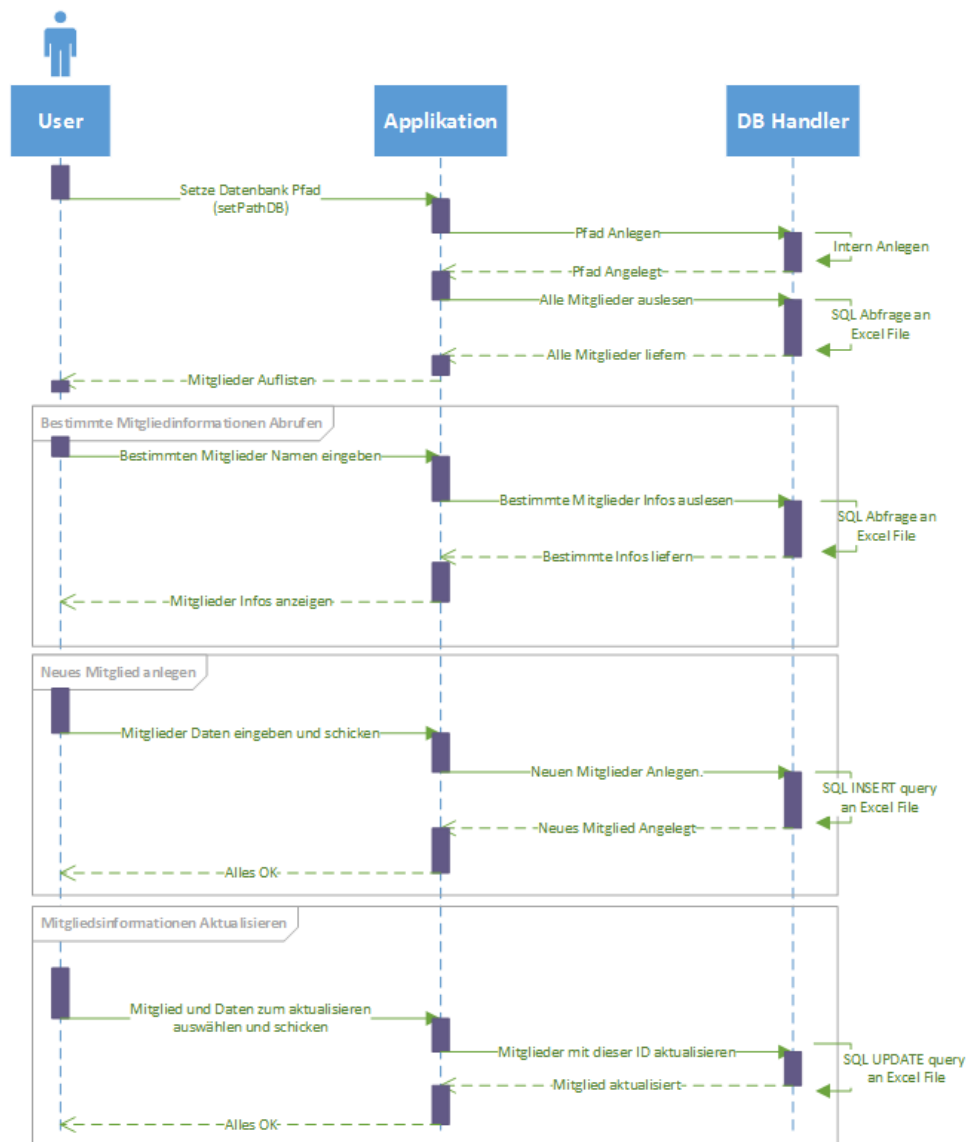


Bild 7: Sequenzdiagramm

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

Bestimmte Mitgliedinformationen Abrufen

Falls der User nach einer Bestimmten Person sucht, gibt er den Namen, Nachnamen, Adresse oder Ort ein und es wird bei jeder Text Änderung eine SQL abfrage geschickt und entsprechende Datasets zurückgeliefert.

Neues Mitglied anlegen

Falls der User ein neues Mitglied registrieren 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 einen Mitglied in der Liste anklickt, kann er die Informationen aktualisieren oder das Mitglied löschen. Beim löschen wird einfach eine NULL update geschickt weil das direkte löschen 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:

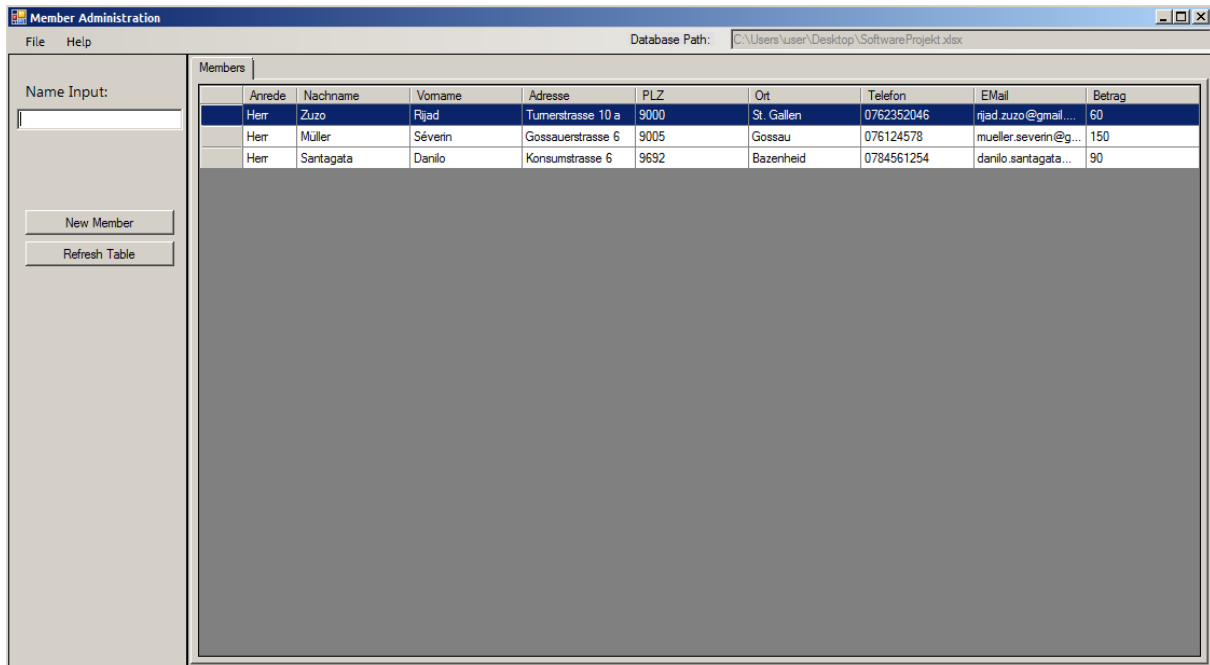
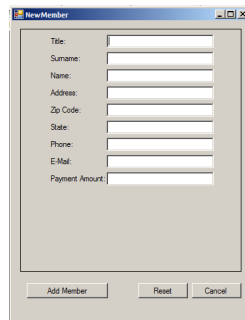


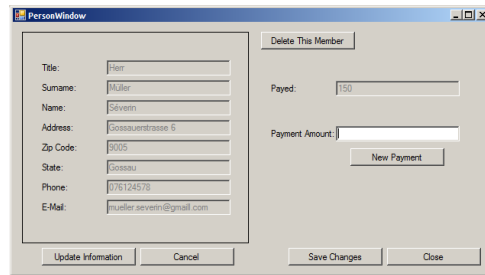
Bild 8: Hauptfenster

- 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 "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 (Payed) 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 welche alle Interaktion mit der Datenbank, in unserem Fall mit dem Excel File, übernimmt.

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

Da Zugriffskollision ein grosses Problem bei Verteilten Dateien ist, haben wir sichergestellt das nach jeder Interaktion mit dem Excel File die Verbindung unterbrochen wird. Der Verbindungsmanager merkt ob mit dem Excel File kommuniziert wird und solange hält er die Verbindung aufrecht.

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

6 Testen

GUI Tests

- | | | |
|-----------------------|--|---|
| 1. Filechooser Test | Im Menu Strip -> File -> Choose Excel File
Ein File Chooser Menu wird aufgerufen. | ✓ |
| 2. Excel File Auswahl | Im File Chooser die gewünschte Datei auswählen und auf OK drücken.
FileChooser wird geschlossen und die Tabelle mit Daten gefüllt | ✓ |

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. ✓
- 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