

PA 2023  
Abschlussarbeit

Projekt-Analyse

Alain Hoch

blue office AG

Turbistrasse 10

6280 Hochdorf

# Inhaltsverzeichnis

[1 Inhaltsverzeichnis 2](#_Toc127279712)

[2 Vorwort 4](#_Toc127279713)

[3 Organisation der Arbeitsergebnisse 4](#_Toc127279714)

[4 Umfeld und Ablauf 5](#_Toc127279715)

[4.1 Aufgabenstellung 5](#_Toc127279716)

[4.1.1 Titel der Arbeit 5](#_Toc127279717)

[4.1.2 Ausgangslage 5](#_Toc127279718)

[4.1.3 Detaillierte Aufgabenstellung 5](#_Toc127279719)

[4.1.4 Aufbau 6](#_Toc127279720)

[4.1.5 Funktionen 6](#_Toc127279721)

[5 Projektorganisation 7](#_Toc127279722)

[6 Mittel und Methoden 8](#_Toc127279723)

[7 Vorkenntnisse 8](#_Toc127279724)

[8 Vorarbeiten 8](#_Toc127279725)

[9 Arbeiten in den letzten 6 Monaten 8](#_Toc127279726)

[10 Hilfestellung 8](#_Toc127279727)

[11 Zeitplan 10](#_Toc127279728)

[12 Arbeitsjournal 11](#_Toc127279729)

[13 Projekt 18](#_Toc127279730)

[13.1 Zusammenfassung (Kurzfassung) 18](#_Toc127279731)

[13.2 Ausgangslage (kurze Ausgangssituation) 18](#_Toc127279732)

[13.3 Umsetzung 18](#_Toc127279733)

[13.4 Ergebnis 18](#_Toc127279734)

[14 Einleitung 19](#_Toc127279735)

[15 Informieren 20](#_Toc127279736)

[15.1 Ziele der Aufgabenstellung 20](#_Toc127279737)

[15.2 Vorgaben 20](#_Toc127279738)

[15.3 Fragen 21](#_Toc127279739)

[16 Planen 21](#_Toc127279740)

[16.1 Realisierungskonzept 21](#_Toc127279741)

[16.2 Klassen-Diagramm 23](#_Toc127279742)

[16.3 Datenbank-Diagramm 24](#_Toc127279743)

[16.4 Testkonzept 25](#_Toc127279744)

[17 Entscheiden 29](#_Toc127279745)

[17.1 Varianten 29](#_Toc127279746)

[17.2 Entscheid 29](#_Toc127279747)

[18 Realisieren 30](#_Toc127279748)

[18.1 Projektumgebung 30](#_Toc127279749)

[18.2 Datenbank 31](#_Toc127279750)

[18.3 clsAddIn 32](#_Toc127279751)

[18.4 ADR-Control 33](#_Toc127279752)

[18.4.1 InitializeControl 34](#_Toc127279753)

[18.4.2 CurrentObject 35](#_Toc127279754)

[18.4.3 Fill 36](#_Toc127279755)

[18.4.4 Buttons 41](#_Toc127279756)

[18.4.5 boSearchList 41](#_Toc127279757)

[18.4.6 boGrid 42](#_Toc127279758)

[18.4.7 Styles von boGrid 43](#_Toc127279759)

[18.4.8 Laden von RegForm 44](#_Toc127279760)

[18.4.9 Buttons und EventHandlers 44](#_Toc127279761)

[18.4.10 AddBeleg 44](#_Toc127279762)

[18.4.11 AddSR 45](#_Toc127279763)

[18.4.12 Delete Beleg/SR 46](#_Toc127279764)

[18.5 SSV-Control 46](#_Toc127279765)

[19 Kontrollieren 47](#_Toc127279766)

[19.1 Testprotokoll 47](#_Toc127279767)

[19.2 Testbericht 47](#_Toc127279768)

[20 Auswerten 47](#_Toc127279769)

[20.1 Reflexion 47](#_Toc127279770)

[21 Glossar 47](#_Toc127279771)

[22 Abbildungsverzeichnis 47](#_Toc127279772)

[23 Quellenverzeichnis 47](#_Toc127279773)

[24 Anhang 47](#_Toc127279774)

[24.1 Anleitung 47](#_Toc127279775)

[24.2 MSSQL 47](#_Toc127279776)

[24.3 VB.Net 47](#_Toc127279777)

# Vorwort

Diese Dokumentation gehört zur PA von Alain Hoch bei der Firma blue office AG. In dieser Dokumentation ist der Ablauf der Arbeit beschrieben und das Vorgehen dahinter. Die Dokumentation ist in 2 Teile gegliedert.

Im ersten Teil wird die detaillierte Aufgabenstellung und der Ablauf der Arbeit aufgezeigt. Es ist ersichtlich mit welchen Mitteln gearbeitet wurde und welche Vorkenntnisse vorhanden sind.

Im zweiten Teil kommt dann die Projekt Dokumentation in der die eigentliche Arbeit beschrieben wird, was für Probleme aufgetreten sind und wie die Arbeit getestet wurde. Durch die ganze Arbeit hinweg wurde mit der Projektplanungsmethode IPERKA gearbeitet. Die Reflexion und das Fazit stehen am Schluss.

# Organisation der Arbeitsergebnisse

Die Dokumentation sowie die restlichen Dateien, die nicht im Visual Studio bearbeitet werden können, werden bei jedem Zwischenspeichern oder manuell ins OneDrive gespeichert. Für die Office Dateien wird dort automatisch eine Versionierung durchgeführt, welches bei jedem Zwischenspeichern durchgeführt wird. Die Restlichen Daten werden von mir manuell auf GitHub hochgeladen und dort Versionisiert. E ist möglich auf GitHub jeden Upload, einen so genannten «Push», nach zu verfolgen und nach zu schauen was geändert worden ist. Weiter wird Lokal auf dem Computer noch eine Sicherung gemacht. Somit sind die Daten an 3 Verschiedenen Orten gespeichert und gesichert.

# Umfeld und Ablauf

## Aufgabenstellung

### Titel der Arbeit

Projekt-Analyse

### Ausgangslage

Die Firma blue office AG benötigt für Neukunden-Projekte eine Übersicht, bzw. eine Überwachung der Offerierten Dienstleistungsstunden mit den erbrachten Leistungen. Es soll erkennbar sein, ob die erbrachten Dienstleistungen den offerierten Aufwand - aus der Auftragsbestätigung - übersteigen, bevor die Dienstleistungsrechnungen erstellt sind, bzw. die Supportfälle abgeschlossen sind)

### Detaillierte Aufgabenstellung

Hintergrundinformationen:

In den Auftragsbestätigungen sind Arbeitsstunden (Dienstleistungsartikel vom Typ «Leistung») erfasst. In den Service-Anfragen sind die «Bearbeitungszeit-Einträge» der geleisteten Stunden hinterlegt, diese Stunden werden automatisch in die Zeiterfassung geschrieben. Es müssen nun die Stunden aus der Zeiterfassung, die dieser Service-Anfrage zugeordnet sind, ausgewertet werden.

Allgemein:

Es muss ein Add-In für blue office® erstellt werden. Als Grundlage der Erstellung des Add-Ins soll das interne Dokument «Kurzbeschreibung AddIn-Erstellung.pdf» und «Beschreibung AddIn-Möglichkeiten.pdf» verwendet werden. Das Projekt, bzw. die Add-In DLL soll einen sprechenden Namen erhalten. Z.B. boProjektAnalyse.dll. Der Präfix «bo» ist die Abkürzung für «blue office». Aufgabe Adressverwaltung: Add-In erstellen, das in der Adressverwaltung als Control (im Form-Designer) eingefügt werden kann. Es muss eine Tabelle in der Datenbank erstellt werden, die die Zuordnung der Adressen mit den Aufträgen und den Service-Anfragen speichert. Es müssen pro Adresse verschiedene Aufträge und Service-Anfragen hinzugefügt und entfernt werden können. Es dürfen nur Aufträge und Service-Anfragen hinzugefügt werden, die auch der entsprechenden Adresse zugeordnet sind. Alle zugeordneten Aufträge und Service-Anfragen sollen direkt auf dem Control ersichtlich sein. Die Arbeitsstunden aus dem Auftrag (SOLL) sollen nun mit den geleisteten Stunden aus der Zeiterfassung (IST) gegenübergestellt werden. • Summe aller SOLL-Stunden • Summe aller IST-Stunden, diese soll zuzüglich noch pro Verrechnungstyp unterteilt werden in: - Summe aller IST-Stunden «Verrechnen» - Summe aller IST-Stunden «Warten» - Summe aller IST-Stunden «Kulanz» - Summe aller IST-Stunden «Garantie» - Summe aller IST-Stunden «Nicht Verrechnen» Folgendes soll visuell ersichtlich sein (Z.B. rot eingefärbt oder als Prozentanzeige): - IST-Stunden (nur Verrechen+Warten) ggü. SOLL-Stunden «Kundensicht» - IST-Stunden (alle) ggü. SOLL-Stunden «blue office Sicht» Wenn IST-Stunden 90% der SOLL-Stunden erreichen, soll das ebenso visuell ersichtlich sein. Z.B. orange eingefärbt. (Farben ggf. einstellbar) Ein Auftrag, oder Service Anfrage darf nur einmal einer Adresse hinzugefügt werden. Ein Auftrag oder eine Service-Anfrage, soll direkt über die Auflistung(en) geöffnet werden können. Doppelklick und via Kontextmenu Layout: Die Darstellung überlassen wir Alain Hoch. Der SOLL / IST Vergleich muss jedoch gut ersichtlich platziert werden. Das Einfügen von Aufträgen und Service Anfragen soll logisch angeordnet sein. Z.B. via Button oberhalb der entsprechenden Liste und auch via Kontextmenu. Für Auflistungen der Aufträge und Service-Anfragen, müssen die Controls wie «boSearchList» oder «boGrid», aus der blue office API verwendet werden. Bei Auflistung von Aufträgen, muss mindestens die Auftrags-Nr. (A.XXXX), Bezeichnung und Datum aufgeführt werden. Bei Auflistungen von Service-Anfragen, muss mindestens die Service-Nr. (SR.XXXX) und der Titel, Erstellungsdatum aufgeführt werden. Aufgabe Service-Anfrage: Hier muss ein Control erstellt werden das im Form-Designer bei den Service-Anfragen hinterlegt werden kann. Hierbei geht es darum, wenn eine Service-Anfrage geöffnet ist, dass man die SOLL/IST Stunden aller dazugehörigen Aufträge und Zeiterfassungs-Einträge sieht. Es kann sehr minimal gehalten werden und soll nur die entsprechenden Stunden ausweisen. Ausreichend wäre auch eine Prozentanzeige von IST-Stunden (nur Verrechen+Warten) >= SOLL-Stunden. Auch hier überlassen wir Alain Hoch die Darstellung. Das «Control» soll aber nicht zu gross sein und die Kennzahl «Prozentanzeige» enthalten. Evtl. auch mit Farben arbeiten bei kritischen Werten. Benutzerhandbuch: Es soll ein Benutzerhandbuch für blue office Mitarbeiter erstellt werden.

**Optional:** • Im Control der Service-Anfrage soll die Möglichkeit bestehen, das Control der Adressverwaltung aufzurufen. Das soll in einem Dialog (eigener Form) geschehen, das die Möglichkeit besteht weiter Aufträge oder Service-Anfragen hinzuzufügen.

**Prio 1.** • Verwaiste Einträge «Leichen» Wenn z.B. ein Auftrag oder eine Service-Anfrage gelöscht wird, weiss das Add-In nichts davon. D.H. es gibt in der Tabelle Einträge, die keinen Treffer mehr haben. Diese könnten bereinigt werden, wenn man die Adresse wechselt.

**Prio 2.** • (Löschen ist bei blue office nicht erlaubt, wäre aber bei anderen Kunden denkbar)

### Aufbau

Die Grafische Darstellung ist mir selber überlassen, jedoch für die Auflistung der Daten und der Belege und Service Requests, muss ich eine boSearchList verwenden (für Belege und SR’s) und für die Daten der SR’s werde ich ein boGrid verwenden, das sind alles Listen und Grids, welche von unserem API zur Verfügung gestellt werden.

### Funktionen

Das Add-In umfasst folgende Funktionen

Datenerfassung :

* Belege können via Dialog (bo) in die boSearchList hinzugefügt werden. (Button oder ContextualMenu)
* ServiceRequests können via Dialog (bo) in die boSearchList hinzugefügt werden. (Button oder ContextualMenu)
* Die Belege und SR’s können aus der boSL entfernt werden.(Button)

Datenmanipulation:

* Belege und SR’s können direkt aus der boSL heraus geöffnet und manipuliert werden (bo-API) via Doppelklick

Datenausgabe:

* boSL als Übersicht, welche Belege und SR’s im Add-In hinzugefügt wurden.
* boGrid als Übersicht für die Kalkulationen.
* Fortschritt Balken als Fortschrittanzeige (Ist-/Soll-Wert vergleich)

# Projektorganisation

**Lehrbetrieb und Durchführungsort:**

blue office AG

Turbistrasse 10

6028 Hochdorf

041 911 07 11

info@blue-office.ch

**Kandidat:**

Alain Hoch

Schulhausstrasse 15

6030 Ebikon

079 613 46 89

alain.hoch@hotmail.com

**Berufsbildner/Lehrfirma:**

Frank Melber

Bénédict-Schule Luzern

Inseliquai 12B

6005 Luzern

**Verantwortliche Fachkraft:**

Marcel Ruckstuhl

blue office AG

Turbistrasse 10

6028 Hochdorf

m.ruckstuhl@blue-office.ch

**Hauptexperte:**

Michael Estermann

michael@estermann.family

+41 766 19 46 (G)

bbv Software Service AG

Blumenrain 10

6002 Luzern

**Nebenexperte:**

Felix Rohrer

Felix.rohrer@leuchterag.ch

+41 226 50 34

Leuchter IT Solution AG

Winkelriedstrasse 45

6003 Luzern

# Mittel und Methoden

- Programmiersprache: VB.NET

- Auflistungen (Grids): Wenn immer möglich Listen aus der blue office API (blueoffice.controls.dll) verwenden. "boSearchList" und "boGrid"

- Ressourcen wie Icons. Falls entsprechende Grafik vorhanden, immer diese der blue office API (blueoffice.common.dll) verwenden - Datenbankzugriff: Erfolgt immer über die blue office API. (blueoffice.common.dll und blueoffice.DAL.dll)

# Vorkenntnisse

- blue office Klassen (API)

- Erstellung von blue office Add-In

- Erfahrungen in der Applikation blue office

- Kenntnisse der internen Abläufe

# Vorarbeiten

- Raster für Dokumentation

- Vorlage für Anleitung

# Arbeiten in den letzten 6 Monaten

- e-Shop Import, Code von VB6 zu VB.NET "übersetzt".

- GDPDU Export

- Datenimport aus andere Datenbanken

# Hilfestellung

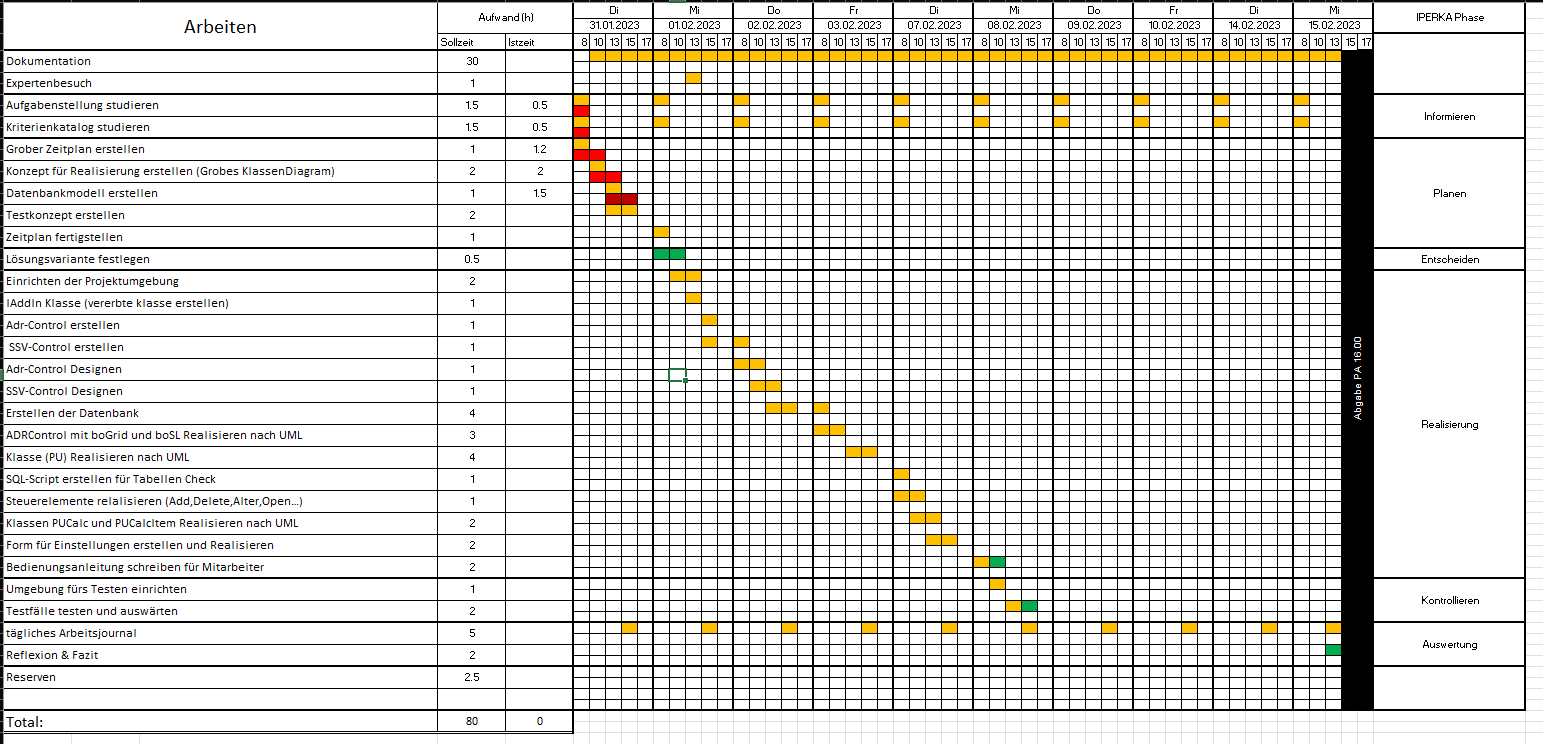
Anselm Schacher:

* DB- Relationen und Darstellung von diesen durch Anselm.

Marcel Ruckstuhl:

* Hilfestellung durch Marcel, da sich Control verzogen hatte.

# Zeitplan



a

# Arbeitsjournal

Tag 1: 31.01.23

|  |  |
| --- | --- |
| Tätigkeiten | Aufgabenstellungen Studiert, Kriterien-Katalog studiert, Zeitplan erstellt, Klassen- DB-Diagramm erstellt, Testkonzept erstellt |
| Erreichte Ziele | Für heute habe ich alles erreicht, was ich mir vorgenommen habe, ich wollte die Planung heute so gut wie möglich vorbereiten.   * Zeitplanung * Diagramme * Testkonzepte |
| Probleme | Ich musste etwas genauer hinschauen beim Datenbank-Diagramm und beim Klassen-Diagramm, diese waren nicht einfach zu erstellen. Jedoch denke ich habe ich einen guten Weg gefunden diese Grafisch dar zu stellen. |
| Hilfestellung | Marcel hatte mir einen alten boMandanten gegeben und geholfen zu installieren, um für Morgen gewappnet zu sein, wenn es an die DB geht. |
| Ausserplanmässige Arbeiten | E-Mail beantwortet. |
| Reflexion | Im Ganzen bin ich gut in die PA gestartet, ich konnte mich super konzentrieren und war zielstrebig auf dem Weg unterwegs, welcher ich mir heute geplant habe, bislang läuft es gut.  Wie gesagt, bei den Diagrammen tat ich mich etwas schwerer als gedacht, jedoch bin ich mit dem Ergebnis zufrieden.  Den Zeitplan musste ich zwischendurch wieder anpassen, da ich 1-2 Fehler gefunden gehabt habe. Diese waren jedoch schnell behoben und störten mich nicht weiter an der PA. |

Tag 2 01.02.23

|  |  |
| --- | --- |
| Tätigkeit | Datenbank-Diagramm neu erstellt, Zeitplan vervollständigt, Lösungsvariante Festgelegt, Projekt Umgebung eingerichtet, Alle Klassen erstellt jedoch nur bei ctlADR und ctlSSV implementieren angefangen. Experten Besuch |
| Erreichte Ziele | Alles, was ich mir für heute vorgenommen hatte. Unteranderem wurde der Meilenstein «Lösungsvariante festlegen» erreicht. |
| Probleme | Ich hatte ein Problem mit meinem Datenbank-Diagramm, es störte mich gestern Abend schon, wie ich es gemacht hatte, daher habe ich es heute Morgen mit Fremdhilfe von Anselm neu erstellt. |
| Hilfestellung | Darstellung des DB-Diagramms und Besprechung wie die Tabellen verbunden werden sollen durch Anselm. |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Da mich das ganze gestört hatte, wie ich das DB-Diagramm gemacht hatte, habe ich es erneut erstellt. Ich bin froh habe ich das gmacht, somit hatte ich die Relationen auch besser verstanden, wie ich sie machen sollte. Weiter verlief der Tag nach Plan, das ganze hatte mich nicht weit nach hinten geworfen, im Gegenteil ich lag sehr gut im Zeitrahmen und konnte so etwas vorarbeiten. Ich habe am Schluss noch Par zielen an der Dokumentation gearbeitet und bin mit dem Heutigen Tag sehr zufrieden.  Zum Experten-Besuch war ich etwas nervös, jedoch kam die Entspannung sehr schnell als ich merkte, dass das alles nicht so wild ist.  Weiter werde ich morgen versuchen weiter konzentriert an der Arbeit zu sein, um im Zeitplan zu bleiben. |

Tag 3 02.02.23

|  |  |
| --- | --- |
| Tätigkeit | SSV\_Control Implementierung angefangen, Datenbank erstellt, ADRControl mit boGrid und boSL Realisiert. |
| Erreichte Ziele | Die Heutigen Ziele, die Tabellen in der Datenbank und auch die SearchList und das Grid realisiert zu haben, wurden erreicht. |
| Probleme | Leider hatte ich grosse Probleme mit meinem GitHub, ich wollte einen Push machen, jedoch funktionierte dieser nicht. Ich versuchte dann das Ganze auf einem separatem Branch (Main mit 2.Branch) zu mergen jedoch hat dies auch nicht funktioniert, ich habe dann Git auf meinem Lokalen Ordner gelöscht und alles frisch initialisiert und dann hat es das ganze Repository überschrieben. Ich hatte dann nur noch 1 Commit und alle daten waren weg. Ich war jedoch vorbereitet und hatte alles noch auf OneDrive und Lokal auf dem PC gespeichert. Daher hatte ich kein Datenverlust.  Jetzt läuft alles wieder.  Als ich das ganze geregelt hatte, wollte ich die ProjektAnalyse anschauen im blueoffice, jedoch war dies falsch formatiert. Ich konnte es mit Marcel nach 40 Min herausfinden, was der Fehler war. |
| Hilfestellung | Keine |
| Ausserplanmässige Arbeiten | Reparieren von GitHub, Reparieren von ctlADR |
| Reflexion | Der Morgen verlief ohne Probleme, ich kam gut vorwärts und konnte alles umsetzen so wie ich das wollte. Jedoch am Nachmittag, war vieles falsch gelaufen. Wie oben in der Sektion «Probleme» beschreiben. |

Tag 4 03.02.23

|  |  |
| --- | --- |
| Tätigkeit | SSVControl fertig implementiert, ADRControl auch Fertig implementiert. Folgende Klassen erstellt: PUCalc, PUCalcItem diese Realisiert nach UML. Und frmSettings angefangen. |
| Erreichte Ziele | Ich wollte heute mit allen Kalkulationen fertig werden, welche gemacht werden wie auch mit den Buttons. |
| Probleme | Ich hatte bei der Kalkulation von den Daten einige Probleme, dass am Anfang nicht alles richtig angezeigt wurde. Auch bei den Rows im Grid hatte es mir beim Refresh nicht die vorherigen Rows gelöscht. Das hatte ich aber schnell bereinigt. |
| Hilfestellung | Keine |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Heute ist viel gegangen, die Ziele waren sehr gross, jedoch konnte ich alles erledigen, was ich mir vorgenommen hatte. Ich war froh, dass die Fehler von gestern Abend nicht mehr Probleme produzierten. Somit konnte ich fokussiert weiterarbeiten. Die Umsetzung alles Kalkulationen benötigte viel Zeit, jedoch habe ich das vorgängig kalkuliert und passte gut in die geplante Zeit. Weiter hatte ich jedoch die Implementierung der Buttons etwas unterschätzt und daher mehr Zeit gebraucht als geplant. Fazit der ersten Woche fällt positiv aus. Ich konnte gut vorankommen und sehe wie weit ich schon bin. Das macht mich glücklich, dass ich bis jetzt keine weiteren grossen Problemen entgegenstehen musste.  Ich hoffe es verläuft nächste Woche gleich.  Für nächste Woche kann ich genau nach dem Zeitplan weitermachen. und den IPERKA- Schritt Realisierung, abschliessen |

Tag 5 07.02.23

|  |  |
| --- | --- |
| Tätigkeit | Form für Einstellungen erstellt, Bedienungsanleitung erstellt für Mitarbeiter  Bugs behoben, Contextual Menü hinzugefügt |
| Erreichte Ziele | Form für Einstellungen, Bedienungsanleitung, |
| Probleme | Ich hatte einige Bugs im Umgang mit dem Add-In, welche ich bereinigen musste. Unteranderem, dass die Rows vertauscht waren und unteranderem dass beim neu erstellen eines SR dieser auf eine Exception geführt hat, da nicht auf dr is nothing abgefragt wurde. Jetzt funktioniert jedoch alles. Beim ContextualMenu wollte ich das von unserer API verwenden, und hatte Probleme, dies zu implementieren. Anselm versucht mir zu helfen, hatte jedoch auch Schwierigkeiten. Als ich anschliessend zu Marcel ging und er meinte, dass ich dies so nicht implementieren könne, ich müsse ein eigenes machen, was ich dann auch anschliessend gemacht habe. |
| Hilfestellung | ContextualMenu durch Anselm und Marcel |
| Ausserplanmässige Arbeiten | Bug Fixing, ContextualMenu |
| Reflexion | Ich konnte am Morgen gut einsteigen wieder nach dem Wochenende. Ich wusste direkt, wo ich weiter machen musste und das tat ich auch. Daher war das Realisieren der Einstellungs-Form keine grosse Sache. Jedoch war ich lange unzufrieden mit dem Style, so dass ich etwas mehr als geplant brauchte um alles fertig zu stellen.  Das Erstellen der Bedienungsanleitung war Fleissarbeit, also kein Problem. Ich hatte zwischendurch auch an dem ContextualMenu gearbeitet, da ich dies nicht richtig umsetzen konnte und es später erneut versuchen wollte, war jetzt ein guter Zeitpunkt, am Schluss vom Projekt, wo ich noch ein bischen Zeit habe, bevor ich mit dem Testen beginne.  Als ich schon etwas vorgezogen einige Daten für die Testfälle morgen erstellen wollte, ist mir ein Bug aufgefallen, bei welchem ich einen dummen Fehler gemacht habe. Jedoch eine kurze Absprache mit Marcel lüftete das Problem und ich konnte weiter machen. Es war rein nur eine Abfrage von dem dr Objekt, welches eine DataRow ist. Falls dies Nothing ist wirft der Editor eine Exceptio. |

Tag 6 08.02.23

|  |  |
| --- | --- |
| Tätigkeit | Testumgebung fertig einrichten, Testfälle testen und auswerten (falls möglich Fehler gleich fixen) |
| Erreichte Ziele | Abschluss von Realisierungs Phase |
| Probleme | Bugs waren mehr als ich gedacht hatte |
| Hilfestellung | Keine |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Ich konnte ja gestern schon etwas anfangen mit dem vorbereiten der Entwicklerumgebung.  Ich hatte einen neuen Mandanten verwendet, welcher noch nie mit dem AddIn zu tun hatte. Das war gut, denn ich bemerkte gleich, dass mir ein Fehler eingeschlichen ist im Tabellen Check.  Die ID war nicht auf AutIncremetn eingestellt, daher hat es mir einen Fehler beim Inserten gegeben. Dies hatte ich jedoch schnell bereinigt gehabt.  Zusätzlich ist mir aufgefallen, dass beim hinzufügen von Rapporten auf den verschiedenen Verechnungsarten unterschiedlich die Stunden angerechnet werden. Ich nehme an, dies ist nur auf diesen Mandanten spezifisch denn auf anderen hatte ich das Problem nicht.  Sonst verlief alles i.O ich konnte gut vorwärts kommen und ich Konnte meine Ziele erreichen, denn mit dem Abschluss aller Testfälle und dem bereinigen aller Fehler erreichte ich den Meilenstein Realisieren in meiner Zeitplanung und das freute mich sehr! |

Tag 7 09.02.23

|  |  |
| --- | --- |
| Tätigkeit | Bug Fixes, Dokumentation (Bearbeitung von vorherigen Kapiteln, momentaner Stand: IPERKA: Entscheidung. |
| Erreichte Ziele | Ich wollte heute einige Bugs, oder Designs noch geändert haben, welche ich noch als unsauber erachtet habe. Und natürlich bei der Dokumentation wie in der Planung definiert weitermachen. Ziel war es bis zur Phase Entscheidung abzuschliessen. |
| Probleme | Heute war ein reibungsloser Tag. Ich konnte gut voran kommen. Das einzige was ich nicht erreichen konnte, war die Phase Entscheidung abzuschliessen, diese ist jedoch nicht all zu gross. Daher hat dies morgen auch noch Zeit. Ein Entwurf wurde schon geschrieben. |
| Hilfestellung | Keine |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Der heutige Tag war schnell vorbei gegangen, ich konnte gut arbeiten und mich konzentrieren.  Ich hate gestern Abend noch mit dem «Kunden» Moritz mein Chef ein kurzes Gespräch und fragte ihm nach seiner Meinung nach dem Aussehen, ob alles so für ihn stimmen würde. Er Wollte noch paar Sachen Geändert haben und dies habe ich heute noch umgesetzt. Unteranderem war das, dass er gerne den Ist und die KundenSicht beieinander hat um es besser vergleichen zu können. Dies ging jedoch sehr gut und schnell vorbei. Mir selber sind auch noch ein paar Verbesserungen aufgefallen,we lche ich noch verbessert habe. (Progressbar in SSV beschrieben und % Farbig gemacht, Kundensicht wie IST\_wert auch dynamisch Farbig gestaltet. )Weiter hatte ich dann heute mit den Entwürfen weitergemacht, welche ich wärend der Realisierung oder der Planung schon geschrieben habe, überarbeitet.  Ich konnte mein selber gestecktes Ziel nicht ganz erreichen, jedoch ist es nicht so schlimm, ich bin schon allgemein weiter als selber geplant und das ist ein gutes Gefühl. |

Tag 8 10.02.23

|  |  |
| --- | --- |
| Tätigkeit | Dokumentation geschrieben Absatz Entscheidung und Realisation |
| Erreichte Ziele | Entscheidung abgeschlossen |
| Probleme | Total den Umfang von dem Absatz Realisation unterschätzt |
| Hilfestellung | Keine |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Ich hatte den Abschnitt Realisation total unterschätzt, jedoch denke ich, wird das kein Problem geben, denn ich bin schon relativ weit mit der Dokumentation. Dienstag nächste Woche werde ich sicherlich fertig mit der Realisation dann kann ich noch die Reflexion schreiben und alles überprüfen auf korrekte Schreibweise usw. Wie verlief der heutige Tag ?  Es ging mal Berg auf mal Berg ab, die Konzentration wird immer weniger, ich bin froh ist jetzt Wochenende und ich kann eine Pause machen. Nächsten Dienstag sitze ich dann wieder mit vollem Elan hier und mache weiter.  Ich habe Heute etwas früher Feierabend gemacht: 16.00 Uhr |

Tag 9 (Datum)

|  |  |
| --- | --- |
| Tätigkeit | Realisieren Phase weiterschreiben |
| Erreichte Ziele | * Ich wollte heute mit dem Realsieren fertig werden jedoch ist das nicht ganz fertig geworden. |
| Probleme | boGridSR hatte sich nach einer Änderung am Freitag vollkommen verzogen. Ich hatte die Änderung am Freitag nicht bemerkt gehabt, da ich noch im Release gewesen bin zum Kompilieren, daher hatte er es nicht übernommen im Programm. Und Heute als ich auf Debugging wechselte hatte es sich verzogen und ich musste mich auf die Suche machen. |
| Hilfestellung | Kurze Hilfestellung von Marcel und Anselm, Jedoch war der Fehler schnell gefunden. |
| Ausserplanmässige Arbeiten | Keine |
| Reflexion | Der heutige Tag war etwas schleppend am Morgen, ich konnte nicht gut vorwärts kommen. Jedoch verlief der Nachmittag einiges besser. Jedoch hatte dies zur Folge, dass ich mein selber gestecktes Ziel die IPERKA Phase Realisieren in der Dokumentation abzuschliessen nicht fertig gebracht habe. Ich bin kurz davor, jedoch nicht ganz. Das Nervt mich ein wenig jedoch sollte morgen noch Zeit sein alles rechtzeitig abzuschliessen. Tätigkeiten für Morgen ABSCHLUSSTAG:   * SSV dokumentieren, * Kontrolle Dokumentieren (Testfälle, TestRapporte, DetailBeschreibung) * Reflexion schreiben * Nochmals alles überlesen und kontrollieren. * Mitarbeiterbeschreibung nochmals anpassen auf letzte Änderungen. |

Tag 10 (Datum)

|  |  |
| --- | --- |
| Tätigkeit |  |
| Erreichte Ziele |  |
| Probleme |  |
| Hilfestellung |  |
| Ausserplanmässige Arbeiten |  |
| Reflexion |  |

# Projekt

## Zusammenfassung (Kurzfassung)

Die Kurzfassung der Dokumentation richtet sich an Leser mit Fachwissen in der Informatik und vermittelt einen ersten Überblick, welcher zur Erleichterung diesen soll, um die Arbeit und deren Inhalt verständlicher zu machen.

## Ausgangslage (kurze Ausgangssituation)

Gegenwärtig ist es so, dass unsere Aufträge, welche Dienstleistungen beinhalten keine Übersicht bieten, um zu sehen, wie viele Stunden einer Pauschal verrechnete Dienstleistung auf diesem Auftrag verbraucht wurden. Die Arbeit wird auf sogenannten ServiceRequests rapportiert. Diese wiederum werden dann von Hand verglichen mit den pauschalen Stunden im Auftrag. Um das zuverlässiger und übersichtlicher zu gestalten, wird eine Übersicht entwickelt, welche nötige Daten und Vergleiche zwischen pauschalen Soll- und wirklichen IST-Stunden aufzeigt.

## Umsetzung

Die Übersicht wird als Add-In in der Sprache VB.Net gestaltet. Diese wird mit WindowsForms gestaltet, und im Code behind liegt der Code. Einzelne Komponenten reagieren auf Events, andere auf Klassenaufrufe aus der API. Andere werden direkt aufgerufen. Jedoch grundsätzlich wird das Ganze auf 3 Schichten aufgebaut, einerseits die Add-In Vererbung, die Controls und die direkte Beziehungen neuerstellten Klassen eigens für diese Übersicht entwickelt.

Die Controls stellen anhand von WindowsForms visuell alles dar. Im Codebehin wird alles benötigte darauf vererbt vom IAddInControl- und von dem IFrameControlMenu Interface.  
Diese stellen alle nötigen Aktionen und Funktionen zur Verfügung, welche benötigt werden um das Add-In umzusetzen. Der Datenbankzugriff wird durch unsere eigen API-Funktionen durchgeführt.

## Ergebnis

In der Projekt-Analyse ist es möglich einen Beleg mit Leistungen hinzuzufügen. Dieser Wird dann in der Linken liste (boSL) gelistet und Sortiert nach Nr. Dabei sind keine Dopplungen möglich. Die ServiceRequests werden auch in der linken Liste gelistet und Sortiert wie die Aufträge. Das Hinzufügen, Löschen, Aktualisieren und Einstellungen vornehmen ist über Buttons direkt über der boSL gestaltet. Diese können anhand von einem ContextualMenu auch mit einem Rechtsklick ausgeführt werden. Sobald ein Auftrag in die boSL hinzugefügt wird, zeigt auf der rechten Seite das boGrid die Sollstunden an. Diese werden von den einzelnen Leistungen im Auftrag ausgelesen und ergeben als Summe den Soll-Wert. Mit den hinzugefügten SR’s und deren Rapporten, werden die Stunden gerechnet, welche in verschiedene Kategorien fallen:

Verrechnet, Warten, Kulanz, Garantie und nicht Verrechnet.

Weiter werden nun mehrere Werte aus diesen Kategorien gerechnet.   
Kundensicht und IST-Wert.

Die Kundensicht besteht aus den Werten Verrechnet + Warten und IST-Wert sind alle Kategorien zusammengerechnet. Das Zeigt einerseits auf, was der Kunde am Schluss auf der Abrechnung sehen wird und auch für uns als Betrieb wie viele Stunden wir Pauschal gerechnet haben und wie viele wir doch effektiv gebraucht haben.

Um das auch grafisch etwas dar zu stellen wird der Soll / Ist vergleich anhand einer Progressbar prozentual dargestellt. In den ServiceRequests wird die Kundenansicht dargestellt sprich Soll/ Kundensicht in Prozent.

# Einleitung

In diesem Kapitel beschäftige ich mich mit dem Aufbau des Projekts. Das Projekt ist nach der IPERKA-Methode gebaut, Damit einen gegliederten Ablauf im Projekt ersichtlich ist.   
Die IPERKA-Methode umfasst 6 «Phasen»:

Informieren:

Der Auftrag und dessen detaillierte Aufgabenstellung werden so gut wie möglich studiert und geklärt. Die Informationen werden analysiert und nach Wichtigkeit eingestuft. Wichtige Punkte werden so früh wie möglich erkannt und allfällige Fragen geklärt. Das ist wichtig um einen möglichst genauen Zeitplan zu gestalten und auch um zu verstehen was genau der Auftrag ist und was verlangt wird.

Planen:

Mit einer präzisen Planung wird ein strukturierter und guten Projektablauf garantiert. Dieser zeigt auf wo grössere Schwierigkeiten liegen und was einfach zum Umsetzen ist. Falls notwendig und auch möglich werden mehrere Lösungswege für komplexere Aufgaben geplant. Es wird ein Testkonzept geplant wie auch ein Klassendiagramm erstellt.

Entscheiden:

Die unterschiedlichen Lösungswege werden miteinander verglichen und davon wird dann der bestmögliche Lösungsweg ausgewählt. Dabei wird abgewogen, welche Ideen sinnvoll sind und welche nicht. Welche können innerhalb der vorgegebenen Zeitdauer umgesetzt werden und welche nicht.

Realisieren:

Wenn der Ablauf klar definiert ist und alle Komponenten Entschieden worden sind, wird mit der Realisierung begonnen. Dia Arbeitsabläufe werden stetig protokoliert und die verwendete Zeit wird als Ist-Wert in die Zeitplanung eingetragen.

Kontrollieren:

Das Endprodukt muss am Schluss getestet werden. Dabei wird auf das Testkonzept bezogen, welches bei der Planung erstellt wurde. Hierbei werden fehlgeschlagene Tests direkt korrigiert oder als Fehler in der Dokumentation deklariert.

Auswerten:

Die Arbeit wird mit einer Reflexion abgeschlossen. Diese handelt von der Arbeit selbst und über die Erfahrungen, welche man während dieser gemacht hat. Dabei wird angeschaut, was gut ging und was noch Verbesserungspotenzial haben sollte. Das hilft für die zukünftigen Projekte nicht die selben Fehler zu machen die schon gemacht wurden.

# Informieren

Die erste Phase in der IPERKA-Methode ist das Informieren.

Mit dem genauen Durchlesen und das Bearbeiten der Detaillierten Aufgabenstellung werden wichtige Punkte bearbeitet, welche später bei der Realisierung umgesetzt werden müssen. Dabei sind Fragen vorgängig zu klären und nötige Zusatzinformationen einzuholen.  
Dabei kann es nicht schaden über mögliche Probleme, Gedanken zu machen.

## Ziele der Aufgabenstellung

Es soll eine Ansicht generiert werden, bei welcher Aufträge mit Leistungen erfasst werden können. Dazu werden noch sogenannte Service Requests erfasst. In diesen werden die geleisteten Stunden geschrieben welche erbracht wurden. Für die Stunden gibt es verschiedene Stadien in welche sie gesetzt werden können. (Verrechnet, nicht verrechnet, Warten, Kulanz, Garantie)

Wie die Namen schon verraten, werden nicht immer Beträge verrechnet.   
Um die geleisteten Stunden im Auge behalten zu können, wird nun diese Ansicht generiert. Sie soll eine Übersicht darstellen, in welcher man gleich erkennt wie weit ein Auftrag ist, im Vergleich die im Auftrag erfassten und die schon geleisteten Stunden.

Die einzelnen Werte, von den oben benannten Stadien sollen zusammengefasst aufgezeigt werden und leicht lesbar sein. Dabei soll ein Soll / Ist vergleich gemacht werden, bei welchem Farblich gekennzeichnet wird, ob sich der Vergleich noch im «grünen Bereich» befindet oder nicht mehr.

Zusätzlich soll Grafisch eine Anzeige gestaltet werden, welche eine prozentuale Anzeige vom Soll/ ist Vergleich anzeigt. Dasselbe soll auch in den SR’s ersichtlich sein, einfach wird der Prozentuale Wert die KundenSicht anzeigen (Verrechen+Warten gegenüber Soll)

## Vorgaben

- Die Projekt-Analyse soll als Lizenzfreies Add-In gestaltet werden.

- Das ganze Projekt soll in der Sprache VB.Net Framework 4.8 verfasst werden und unsere bo-API verwenden.

- Es soll eine Tabelle auf der Mandanten-DB im MS SQL gestaltet werden, die die hinzugefügten Belege und SR’s abspeichert.

- Die erstellten Controls sollen über den FormDesigner integrierbar sein und manipuliert werden können.

- Farblich dynamische Veränderung des Ist-Werts, sobald ein gewisser Prozentualer Wert erreicht wurde (vom Benutzer einstellbar)

- Alle verrechnungsarten sollen zusammengerechnet und veranschaulicht werden.

## Fragen

Frage an Marcel Ruckstuhl(31.1.23): Wie genau soll die Umsetzung der DB gestaltet werden.?

Antwort: Eine einzelne Tabelle gestalten, welche alle Belege und SR’s aufnimmt, und dafür um diese miteinander vergleichen zu können eine View gestalten, welche beides beinhält als Union:

Die erstellte Tabelle (PA\_ProjAN) in der Relation 1 zu M mit S\_Service und die erstellte Tabelle(PA\_ProjAn) in der Relation 1 zu M mit boBelegK.

# Planen

Die zweite Phase der IPERKA-Methode ist das Planen.

In dieser Phase wird der Zeitplan umgesetzt und das Testkonzept wie das Realiserungskonzept definiert. Das DatenModel mit den jeweiligen Tabellen und Feldern wird entworfen.

## Realisierungskonzept

Ein neues, leeres Projekt wird als Klassenbibliothek mit .NetFramework 4.8 erstellt und gleich am richtigen Ort abgespeichert. Das Projekt wird so eingestellt, dass es mit unserem eigenen Programm startet zum Debuggen. Die Verweise müssen kontrolliert werden, falls diese nicht stimmen (was sie sicherlich nicht tun) dann diese noch hinzufügen. Benötigt werden sicherlich:

. common

. controls

. DAL

. ERP

. ERP.Adressen

. ERP.Belge

. menu

. Settings

Weiter wird die Datenbank erweitert mit denen vom Daten Modell erstellten Tabellen im MS SQL

Der Zugang darauf wird durch unsere bo-API gewährleistet, welches direkte Abfragen auf die angegebenen Tabellen mit dem dazugegebenen Statements macht.   
Die Realisierung des Add-Ins wird anhand einer internen Anleitung durchgeführt, welche als Vorlage dient. Jedes Add-in ist gleich aufgebaut, das Projekt-Analyse Add-in wird ein Lizenzfreies Add-In.  
Das heisst es müssen keine LizenzKey generiert und eingegeben werden, sondern kann einfach mit der entsprechender .dll Datei hinzugefügt und verwendet werden.

Als Einstiegspunkt für die Programmierung wird die Definition sein, wo in welchem Bereich sich das Add-In hinzufügen soll. Der FormDesigner ist wie eine Tür für weitere Optionen, welche wir ansteuern können. Dort Definieren wir das Add-In für die Adressen und die ServiceRequests

Dort können diese dann in die aktuellen Layouts integriert, manipuliert und verwendet werden.

Im Add-In beinhaltet sind die Controls, ADR und SSV. Auf diesen werden mit WindowsForms die Komponenten zur Verfügung gestellt um die einzelnen Listen darstellen zu können oder Buttons um Events abhandeln zu können oder grafische Darstellungen, welche einen Fortschritt anzeigen.

Für die Darstellung der Liste welche die Aufträge und auch die SR’s aufnehmen soll ist klar nur ein Favorit verwendbar. Die boSL bietet hier di beste Möglichkeiten. Sie sucht in einer definierten Tabelle (viewPA\_ProjAn) und Zeigt angegebene Parameter dar. Unteranderem ein ausgewähltes Bild, NR, Datum, Bezeichnung usw. genau so wie es vorgegeben ist im Auftrag, dieses wird in ein Panel definiert. Um die Daten der hinzugefügten Belege und SR’s darstellen zu können wird auf der Rechten Seite der ProjektAnalyse ein boGrid hinzugefügt. Dieses kann verwendet werden um Werte in einer Tabelle darzustellen.

Ich verwende Sie auch genau so, da sie einfach zum handeln ist und sie die Daten Übersichtlich darstellt.

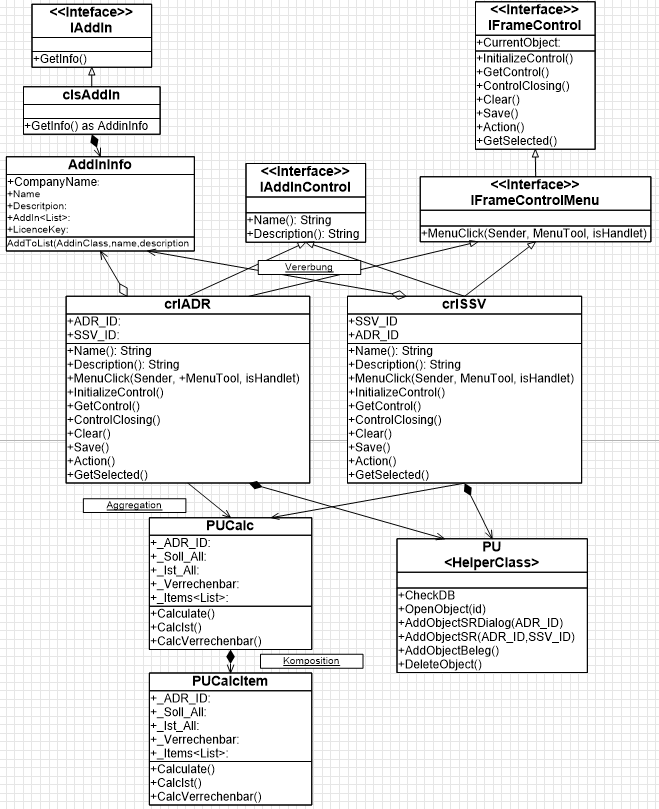
Um Belege und SR’s in die boSL hinzufügen zu können werden Buttons realisiert, welche Belege und SR’s hinzufügen, sie aus der boSL entfernen, das gesamte Control refreshen und die Einstellungen öffnen können.

Die Buttons werden mit einer eigener Klasse versehen, welche die Methoden zur Verfügung stellt.

Die klasse wird direkt aus dem Control heraus beim jeweiligen Buttonklick initiiert und die Methode aufgerufen. (Add, Delete und Refresh)

Die Grafische Darstellung wird als Progressbar integriert. In der ADR im Vergleich mit Soll / Ist und in der SSV im Vergleich KundenSicht / Soll.

## Klassen-Diagramm



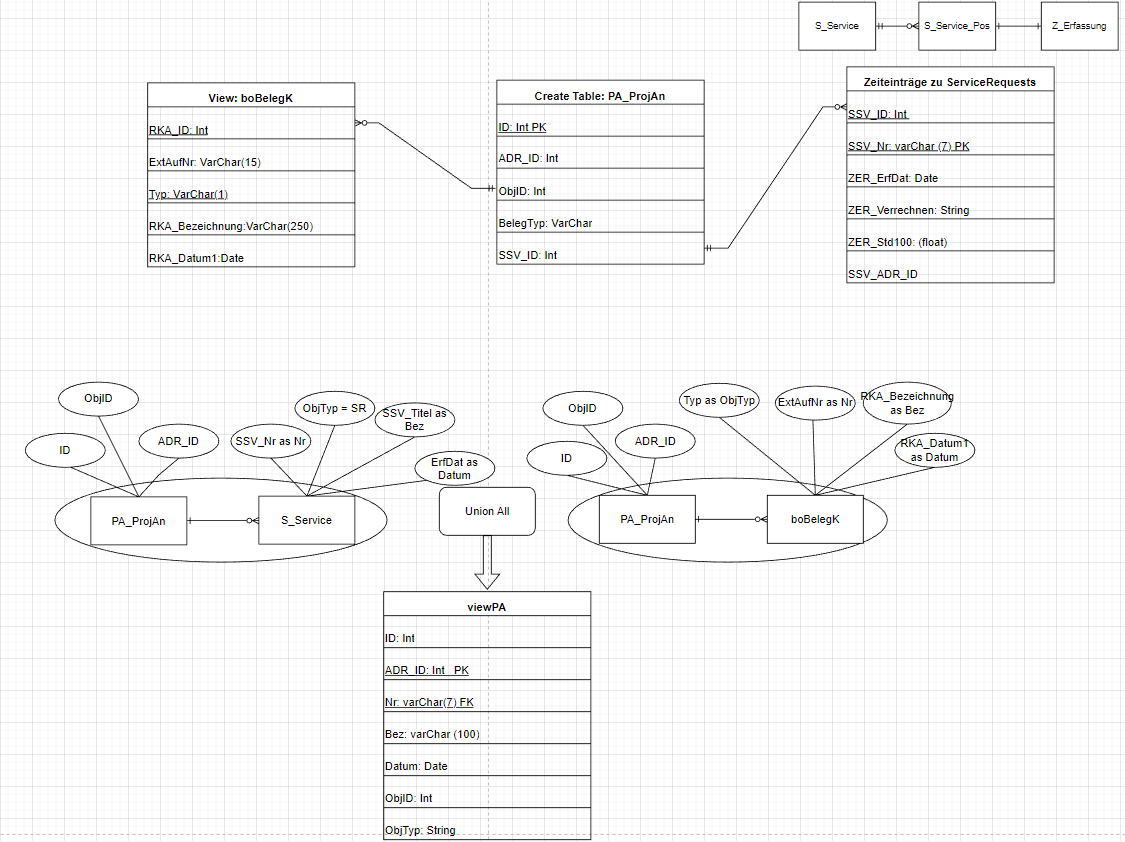
Alle Interfaces und Klassen, welche mit Add-in zu tun haben sind vorgegeben, diese werden durch die Vererbung vorgegeben. Add-InInfo dient als Deklaration in welcher Kategorie ADR/SSV es ersichtlich sein soll im FormDesigner. Alle anderen sind wie hier dargestellt zu realisieren.

## Datenbank-Diagramm

Die Tabelle PA\_ProjAn wird neu erstellt um die hinzugefügten Belege und SR’s abzuspeichern.

Aus der Tabelle PA\_ProjAn wird eine View in einem Union All kreiert. Die eine Seite wird die PA\_ProjAn zusammen mit den SR’s verglichen und anderseits mit den boBelegK

Das hat den Vorteil, dass in dieser Tabelle nur Werte abgespeichert werden, welche auf beiden Seite genau die gleichen Adressen treffen.



## Testkonzept

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 1 |
| **Bezeichnung** | | Add-In lässt sich im FormDesigner einstellen |
| **Situation** | **Von-Zustand** | Noch nicht verwendet |
| **Signal** | Objekt-Einfügen im neuen Register |
| **Eingaben** |  | Auswahl von ProjektAnalyse |
|  | - |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die ProjektAnalyse lässt sich ins Register hinzufügen und manipulieren. |
| **Anzeige** |  |
| **Meldungen** |  |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 2 |
| **Bezeichnung** | | DB-Check |
| **Situation** | **Von-Zustand** | Mandant ohne PA |
| **Signal** | PA wird im FormDesigner hinzugefügt dabei wird direkt ein DBCheck gemacht, welcher einen Check auf die Tabellen PA\_ProjAN und viewPA\_ProjAn macht. |
| **Eingaben** |  | - |
|  | - |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die Tabellen sind im MSSQL vorhanden und Details stimmen (Attribute, PrimaryKey,….) |
| **Anzeige** |  |
| **Meldungen** |  |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 3 |
| **Bezeichnung** | | Add-in lässt sich in der Adresse öffnen |
| **Situation** | **Von-Zustand** | Ungeöffnet (Adressverwaltung) |
| **Signal** | Klick auf Register Projekt Analyse |
| **Eingaben** |  | - |
|  | - |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Das Register geht auf und die ganze Projekt Analyse (boGrid, boSL mit Buttons und Progress bar sind ersichtlich) |
| **Anzeige** | Leere ProjektAnalyse |
| **Meldungen** | Nichts |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 4 |
| **Bezeichnung** | | Befüllen der boSL |
| **Situation** | **Von-Zustand** | Leere Liste |
| **Signal** | Mit Button „Add“ Belege Dialog öffnen und Aufträge einfügen können |
| **Eingaben** |  | Auswahl von Belegen im Dialog |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Wieder zurück auf der Project Analyse |
| **Anzeige** | Belege welche ausgewählt wurden sollen dann angezeigt werden in der boSL. |
| **Meldungen** | Keine. |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 5 |
| **Bezeichnung** | | Füllen von boGrid |
| **Situation** | **Von-Zustand** | Leer /nur Gesamt am anzeigen, diese zeit jedoch nur Null-Werte an. |
| **Signal** | SR-Dialog öffnen |
| **Eingaben** |  | Auswahl der SR; welche von der SQL-Abfrage direkt auf die Adresse abgestimmt werden |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die korrekten Daten sollen nach dem Aktualisieren angezeigt werden. |
| **Anzeige** | Anzeigen der SR’s im boGrid welche vom Benutzer ausgewählt wurden. |
| **Meldungen** | keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 6 |
| **Bezeichnung** | | Daten im boGrid sind korrekt! |
| **Situation** | **Von-Zustand** | Belege und SR’s wurden in der boSL ausgewählt. |
| **Signal** | Aktualisieren mittels button |
| **Eingaben** |  |  |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Daten von (Verrechnen, nichtverrechnen, warten, Kulanz ist, soll) sollen korrekt angezeigt werden |
| **Anzeige** | // |
| **Meldungen** | keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 7 |
| **Bezeichnung** | | Daten sollen noch da sein, nach dem Schliessen der Projekt Analyse. |
| **Situation** | **Von-Zustand** | Geschlossener Projekt Analyse (Adressverwaltung) |
| **Signal** | Öffnen der PA |
| **Eingaben** |  | Keine |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Dass die Daten, welche bim Schliessen von der PA vorhanden waren, immer noch da sind. (Abfrage auf SQL Tabelle und Daten ablesen auf viewPA\_ProjAn) |
| **Anzeige** | // |
| **Meldungen** | Keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 8 |
| **Bezeichnung** | | Vergleich der Zahlen |
| **Situation** | **Von-Zustand** | Daten werden angezeigt |
| **Signal** | Keins |
| **Eingaben** |  | ZeitRapporte von SR und Leistungen der Belege |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die angezeigten Daten sind korrekt ausgewertet. (Linie Gesamt) |
| **Anzeige** | Korrekte Daten |
| **Meldungen** | Keine. |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 9 |
| **Bezeichnung** | | Verfärbung der Ist Zahl bei gewisser % Erreichung |
| **Situation** | **Von-Zustand** | Neutral schwarzer Schrift |
| **Signal** | Sobald ein gewisser (vom Benutzer eingestellter Prozentsatz erreicht wurde, soll sich die IST Zahl farblich verändern (Orange) dieser wird 10% bevor die vom Benutzer eingestellte Rote Verfärbung erreicht wird automatisch vollzogen. |
| **Eingaben** |  | Zeit-Rapporte so das 80% erreicht wird zur Soll-Zeit, (80h von 100h) |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Verfärbte IST-Zahl nach Orange |
| **Anzeige** | Orange Ist-zahl |
| **Meldungen** | Keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 10 |
| **Bezeichnung** | | Verfärbung der Ist Zahl bei gewisser % Erreichung |
| **Situation** | **Von-Zustand** | Orange Schrift. |
| **Signal** | Sobald ein gewisser (vom Benutzer eingestellter Prozentsatz erreicht wurde, soll sich die IST Zahl farblich verändern (Rot) dieser wird angezeigt, sobald der Ist-Wert den angegebenen Richtwert (Einstellungen) erreicht hat) Hier 90% Speich 90h von 100h. |
| **Eingaben** |  | Zeit-Rapporte so das 90% erreicht wird zur Soll-Zeit, (90hvon 100h) |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Verfärbte IST-Zahl nach rot |
| **Anzeige** | rote Ist-zahl |
| **Meldungen** | Keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 11 |
| **Bezeichnung** | | Einstellungen |
| **Situation** | **Von-Zustand** | Ungeöffnet |
| **Signal** | Einstellungs-Knopf betätigt |
| **Eingaben** |  | Auswahl des Prozentsatzes, ab wann die Warnung vollzogen werden sollte. (Verfärbung der Ist-Zahl) |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die Wahl wird beim Radiobutton angenommen und ist nur eine zur gleichen Zeit möglich.  Beim schliessen des Fensters soll die Wahl in die Mandaten spezifische Tabelle geschrieben werden. |
| **Anzeige** | Keine |
| **Meldungen** | Keine |

|  |  |  |
| --- | --- | --- |
| **Testnummer** | | 12 |
| **Bezeichnung** | | Einstellungen |
| **Situation** | **Von-Zustand** | Ungeöffnet |
| **Signal** | Einstellungs-Knopf betätigt |
| **Eingaben** |  | keine |
|  |  |
| **Erwartetes**  **Resultat** | **Ziel-Zustand** | Die Eingaben, welche getätigt wurden im Vorherigen Test sollen von der Mandanten Spezifischen Tabelle abgerufen werden und der spezifische Radiobutton soll markiert sein. |
| **Anzeige** | Der gewählte Radio Button von Test Nr. 9 markiert sein. |
| **Meldungen** | Keine |

# Entscheiden

Die Entscheidung ist die 3.Phase des IPERKA-Models und beinhaltet die Entscheidung, für welche Variante man sich entschieden hat, für welche Lösung man am meisten Vorteile sieht und was der effektivste Weg sein wird.

## Varianten

Wie im Realisierungskonzept ersichtlich werden nicht viele Varianten ermöglicht, jedoch was noch zum entscheiden ist, wie die Einstellungen implementiert werden.

Einerseits können diese in die Optionen des ERP’s implementiert werden mit eigenem Register, dies ist jedoch eher umständlich, da es als Add-In gestaltet wird oder auf der Benutzer Oberfläche direkt, als Button.

## Entscheid

Ich habe mich für die Lösung mit dem Button entschieden. Ich habe schon 4 Buttons, welche ich implementieren werde, einer mehr für die Einstellungen ist einfach logisch und eine saubere Lösung zum implementieren mit einem Formaufruf

Das Projekt ProjektAnalyse wird als Add-In realisiert. Dabei wird intern der Formdesiner angesteuert. Im FormDesigner können Designs gestaltet werden, in welchem Objekte (Controls) hinzugefügt werden können und so eine Übersicht bilden. Das Add-In soll genau diesen ansteuern um eingefügt und manipuliert werden zu können. Der FormDesigner ist eine implementierte Funktion unseres ERP’s.

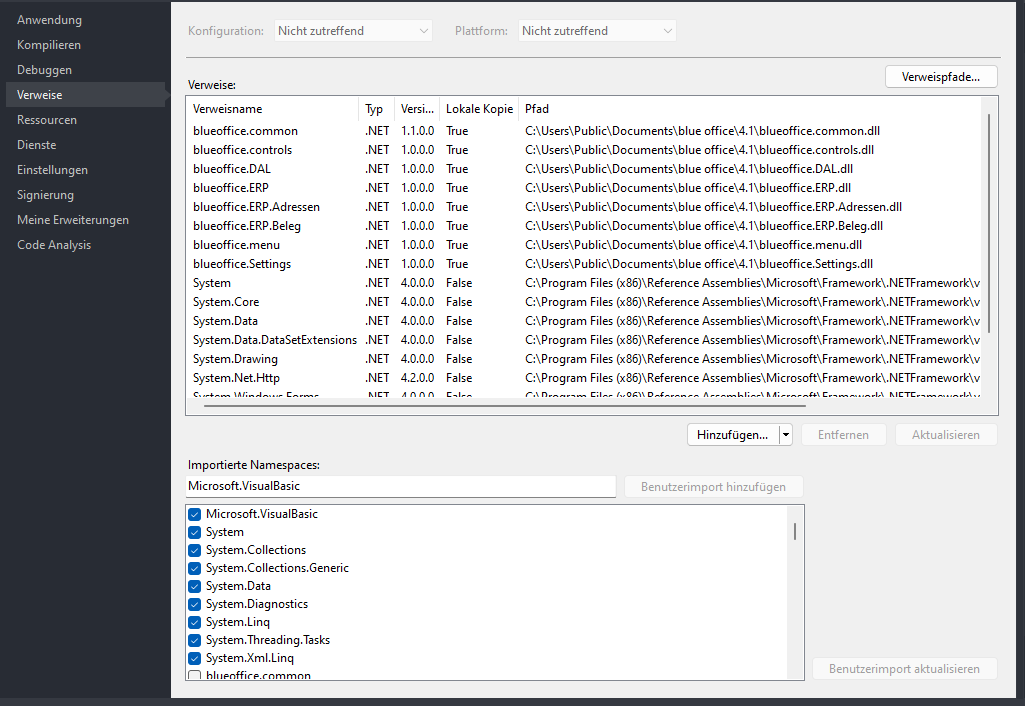
Zusätzlich ist es auch besser für die Prüfung ein Add-In zu gestalten, da dies Lokal programmiert wird und somit eine bessere Prüfungssituation darstellt.

# Realisieren

## Projektumgebung

Die Projektumgebung ist mir wohl bekannt. Wir haben auf unseren Computern Folder, welche die Daten vom ganzen Programm blue office enthalten, das ist unsere Entwicklerumgebung, in welcher wir jegliche Projekte verweisen, und dort hinaus starten wir auch das Kompilieren und Debuggen.

Es enthällt alle benötigten DLL’ s. Daher müssen wir intern im Visual Studio jegliche verweise darauf machen, welche wir wie folgt machen:

Schritt Eins: öffnen von My ProjectEin Bild, das Text enthält.

Automatisch generierte Beschreibung

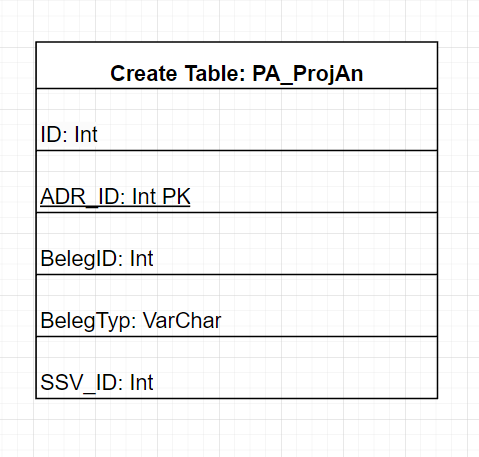


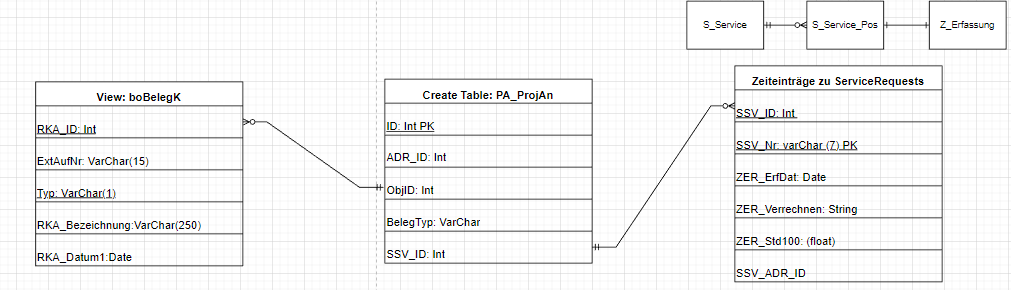
Dabei landen wir hier MyProject und dann auf Verweise.   
Für die müssen wir in den Entwickler Ordner hinein und auf alle benötigten DLL’s verweisen. Somit kann ich später mit den benötigten NameSpaces arbeiten. Ich muss mich darauf achten, da wir diesen Winter auf die Version 4.1 gesprungen sind, dass ich auch nur Verweise auf meinen 4.1 Ordner mache und nicht 4.0

Als nächstes müssen wir beim Register «Debuggen» noch einstellen, dass wir gerne ein externes Programm starten mit dem geschriebenen Code. Somit geben wir unsere blueoffice.exe an, welche sich auch im Entwickler Ordner 4.1 befindet. Ich stelle das immer gleich für beides ein, zum Debuggen und auch für den Release. Da ich den Release später brauchen werde, sobald das Realisieren zu Ende ist.

## Datenbank

Die Tabelle PA\_ProjAn wurde erstellt nach dem Datenbank Model. Die Tabelle dient dazu eingetragenen Belege und SR’s abspeichert, dass bei einem schliessen und wieder öffnen diese auch ersichtlich sind und nicht immer wieder hinzugefügt werden müssen.





Aus der erstellen Tabelle heraus, dem boBelegK und der Zeiterfassung wird eine View Kreiert, welche mit einer Union All gestaltet wird.

Die Union All wird alle Einträge zusammenfassen, welche die gleichen Adressen haben. Dabei wird der Vergleich gemacht dass,

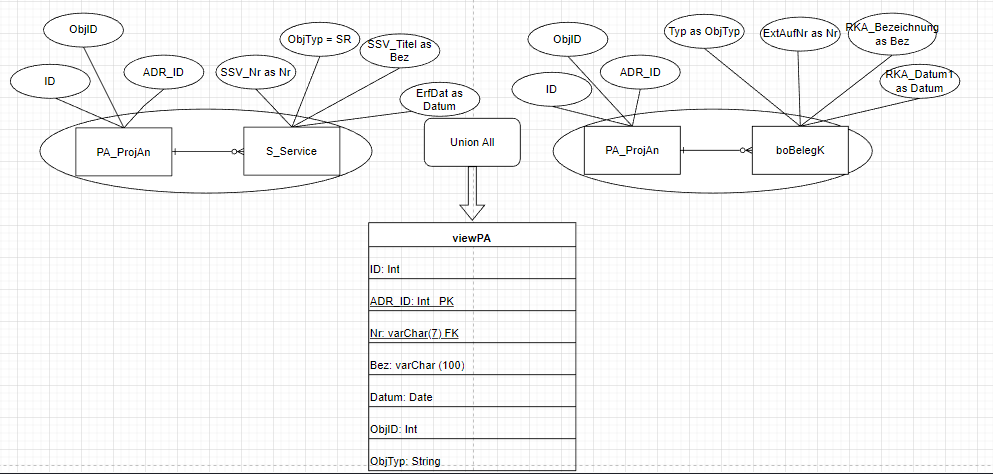
* PA\_ProjAn.SSV\_ID = S\_SERVICE.SSV\_ID ist und
* PA\_ProjAn.BelegID = bo\_BelegK.RKA\_ID And PA\_ProjAn.BelegTyp = bo\_BelegK.Typ

Der erste Vergleich wird auf die Tabelle S\_Service gmacht diese sind die Services, welche auf diese Adresse mit dieser SSV\_Nr passen und beim zweiten Vergleich wird ein Vergleich auf die boBelegK gemacht, welcher auf die ID der Belege und auch auf den Typ vergleicht.

Mit dem Union All ergibt es dann Daten, welche die gleiche Adresse, die darauf gehörigen Belege und die darauf gehörigen ServiceRequests mit den Rapporten haben.

Mit dem können wir nun arbeiten und alles verwenden was wir benötigen.

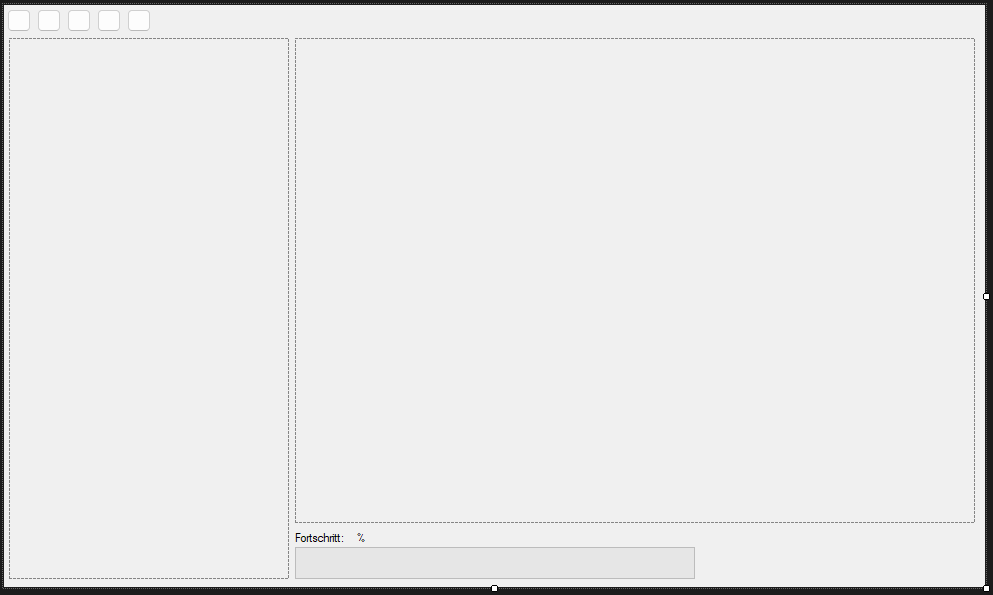
Hier eine Übersicht für den Union All:



## clsAddIn

## ADR-Control

Als erstes habe ich ein Control erstellt, welches ich dann für die AdressVerwaltung verwenden werde. Dies habe ich wie folgt gestaltet:



Das Control implantiert wie im KlassenDiagramm beschrieben zwei Interfaces, welche verschiedene Methoden Definiert. Diese Methoden werden in die Klasse implementiert.  
Die vererbten Methoden sind:

* CurrentObject von der IFrameControl
* ObjectChanged (Verwenden wir nicht)
* PerformAction (verwenden wir nicht)
* MenuClick vom IFrameControlMenu
* InitializeControl von der IFrameControl
* ControlClosing von der IFrameControl
* Clear (Verwenden wir nicht)
* Save (Verwenden wir nicht)
* Action (Verwenden wir nicht)
* Description vom IAddInControl
* GetControl aus der IFrameControl
* GetSelected aus der IFrameControl
* IAddInControl\_Name aus IAddInControl

### InitializeControl

Dier InitializeControl Methode ist die zweite Methode welche beim Aufruf des Controls laufen wird, diese sorgt dafür, dass die boSL und die boGridSR in die Panels eingefügt werden, die allfellige Einstellungen der Spaltenbreiten aus der Reform geladen werden und dass ein DB-Check durcheführt wird. Dieser ist essentiell wichtig, denn ohne die Tabellen funktioniert alles nicht.   
Also habe ich eine Methode geschrieben, welche mit SQL Skripts eine abfrauge auf der Mandanten-DB durchführt über unser boAPI:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dabei wird eine Komposition durchgeführt, welche direkt beim Entstehen des Controls ausgeführt wird. Die DB wird gecheckt auf die Tabellen PA\_ProjAn und die viewPA\_ProjAn.

PA\_ProjAN:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Das Skript wurde erstellt im MS SQL. Das Skript erstellt, falls die Tabelle nicht schon vorhanden ist die PA\_ProjAn Tabelle nach DatenModel.

viewPA\_ProjAn:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Falls die Tabelle PA\_ProjAN nicht vorhanden ist, kann auch diese Tabelle nicht vorhanden sein, daher wird diese auch gleich erstellt ohne neu auf die View abfragen zu gehen.

### CurrentObject

Ein Bild, das Text, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

Das CurrentObject ist ein Property, Properties sind Variable, welche ein Get und Set haben.   
Dabei wird bei dem Get, der aktuelle Wert aus der Variable abgerufen.  
Beim Set wird ein Wert in die Variable geschrieben.

Bei unserem CurrentObject verhält sich das ganze etwas anders. Das CurrentObject wird nach der InitializeControl-Methode aufgerufen. Dabei ist es unser Ziel, das Objekt zu identifizieren. Der eingegebene Wert wird mit Set überprüft, ob er einen Typ ist, welcher in unser DAL-Objekt passt, auf die Tabelle A\_ADRESSEN.   
Falls dies der Fall ist, wird eine Instanz eröffnet mit dem DAL-Objekt auf der A\_ADRESSEN Tabelle und gleich befragt ob es diesen Value sprich das eingegeben Objekt, kennt.   
Falls das ganze Daten hat, dann wird eine von den Klassen-Variablen verwendet um die Momentane Adressen-ID für die ganze Klasse zur Verfügung zu stellen.   
  
Sobald dies geschehen ist, wir die Methode Fill() Ausgeführt.

### Fill

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungDie Fill Methode, ist eines der Herzstücke in der Programmierung.  
In ihr wird als erstes die Methode AufPermFilter() aufgerufen, diese ist in der Klasse weiter unten deklariert und bewirkt, dass sich die boSL nur mit Daten füllen lässt, welche auf die gleiche ADR\_ID lauten wie das momentane Objekt.

boSL.Fill() ist die Methode, um die boSL zu füllen, diese wird aus dem API zur Verfügung gestellt.   
Dabei wird die boSL mit den entsprechenden Werten aus der Datenbank, welche auf der Tabelle viewProj\_An sind und auf den Permanent Filter passen (Adressen-ID des momentanen Objekts.)

Um keine Leere Rows im Grid zu haben, werden diese hier als nächstes auf 1 gesetzt.   
Sprich nur noch die oberste (HeadRow) bleibt bestehen.

##### Die Initialisierung der Klasse PUCalc mittels Aggregation

In der PUCalc Klasse geht es darum, dass die einzelnen Werte, welche wir sehen möchten kalkuliert und auch als Objekte verwendbar sind.

Die PUCalc-Klasse wurde nach dem Klassendiagram realisiert und noch erweitert. Ich habe einzelne Erweiterungen durchgeführt, um die Redundanzen zu verringern und auch alles an einem Ort Kalkuliert zu haben um es nur noch abrufen zu müssen. Wie wird das gemacht ? So:   
  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Als erstes werden in der Klasse viele verschiedene Properties erstellt. (Variablen mit \_am Anfang werden in der Klasse als Properties realisiert.)

Unteranderem Verwenden wir eine Liste, welche die Werte abspeichert als ein Objekt PUCalcItem, dazu jedoch später mehr.

Im Konstruktor der Klasse PUCalc wird die ADR\_ID verlangt, welche wir bei der Instanziierung angegeben haben. Dabei wird eine Klasseneigene Variable den Wert übernehmen und ihn für die ganze Klasse verfügbar machen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die liste wird bei jedem Konstruktor Aufruf neu gebildet, um vorherige Daten, welche sich noch darin befinden könnten, zu eliminieren.

###### Calculate()

Die Methode Calculate, stellt zuerst fest, wie viele Soll Stunden in den Aufträgen kumuliert sind, welche sich in der ProjektAnalyse befinden, auf der jeweiligen Adresse.

Ein SQL-Queue auf der Tabelle PA\_ProjAn wird abgesetzt, welcher eine Summe bildet aus den Werten: RDA\_Anzahl. Dieser Wert stammt von der View bo\_BelegD. Um den Wert generieren zu können wird ein Join verwendet, welcher die Tabellen PA\_ProjAN und bo\_BelegK und bo\_BelegK mit bo\_BelegK miteinander verbindet. Dabei müssen die BelegID und die RKA\_ID gleich sein.   
Um dann auf die View bo\_Beleg\_K zu gelangen, wird erneut einen Join benutz.   
Dieser Verbindet die Views bo\_BelegK und bo\_BelegD über den Typ und die RKA\_ID / RDA\_RKA. Die Verbindungs ID’s der beiden Tabellen zueinander.

Mit der «Where-Klausel» wird die ADR\_ID angegeben also es kommen nur Werte von dieser Adresse.

Am Schluss wird noch mit einem Having (ähnlich wie ein «Where-Klausel» jedoch als Zusatzbedingung deklinierbar) gefiltert, damit die Werte, welche eine Summe bilden sollen auch Leistungen sind und nichts anderes.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Nach dem Absetzten des SQL-Statements wird die DataRow (Summe) überprüft, ob sie nicht Nothing ist. Falls nein, wird die Variable \_totalSoll mit dem Wert aus dem Objekt «dr.item(«Soll»)» festgelegt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Liste der Items wird geleert, um wieder zurückgebliebene Datensätze, welche evtl. nicht sauber gelöscht worden sind zu bereinigen. Als nächster Schritt wird in der Methode Calculate einen weiter SQL-Befehl abgesetzt, Dieser sucht jede SSV\_ID sprich die ID jedes ServiceRequests, welcher sich auf der Tabelle PA\_ProjAN und mit der Adressen-ID übereinstimmt, heraus. Für jeden einzelnen Eintrag wird ein Item in der Liste erstellt und öffnet eine neue Instanz mittels Komposition von der Klasse PUCalcItem.

Initialisierung Klasse PUCalcItem mittels Komposition

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Da sich die Methode Calculate in der Klasse PUCalc im Konstruktor befindet und sich so diese Klasse direkt mit bildet entsteht eine Komposition. Die eine Klasse kann nicht ohne die andere bestehen oder verwendet werden.

Nun auch in dieser Klasse habe ich verschiedene Properties verwendet, jedoch sind dies nur ReadOnly Properties. Ich verwende sie, um später jeden einzelnen Wert aus der Liste Items herauszunehmen und weiter zu verwenden.

Darum sind diese nur ReadOnly. Ich muss die Werte von aussen nur lesen können und nicht überschreiben.

Die Klasse benötigt die SSV\_ID, welche sie bei dem Aufruf in der Klasse PUCalc gleich mitbekommen hat (dr.Item(«SSV\_ID»)).

Im Konstruktor Sub New wird auch in dieser Klasse eine Methode ausgeführt: FillValues()

Die Methode sorgt dafür, dass das Objekt, welches der Liste hinzugefügt wird, auch ihre Werte miterhält.

Als erstes setzt sie einen SQL – Querrey ab, welcher auf 3 Tabellen Joint. Genau so wie in der DB-Beschreibung dargelegt.

Die Tabellen sind: S\_Service, S\_Service\_Pos und Z\_Erfassung.  
Dieses Querrey wird verwendet um die Rapporte in den ServiceRequests zu erhalten, die alle auf die SSV\_ID geschrieben wurden.

Das mache ich wie folgt: Einen inner Join von S\_Service auf die Tabelle S\_Service\_POS, wobei die SSV\_GUID von der S\_Service-Tabelle gleich sein muss wie S\_Service\_POS.SSP\_SSV\_GUID (Verbindungs-ID zwischen diesen Tabellen). Von der S\_Service\_Pos Tabelle geht es über einen weiteren inner Join mit der SSP\_ID, welche gleich sein muss wie die Z\_Erfassung.ZER\_SSP\_ID (Auch wieder eine Verbindungs-ID zwischen diesen Tabellen.)

Nun können wir die Rapporte der einzelnen ServiceRequests analysieren und berechnen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Um die Abfrage nun zu verwerten, wird der Rückgabewert, welchen wir von der Methode GetDataRows erhalten haben, genommen. Mit einer ForEach werden die Item in einer Row von der Tabelle durchlaufen.

Die einzelnen Items werden nun abgefragt. Wir fragen mit einem IF-Statement, ob das Item, welches gerade durch die ForEach läuft, eine «SSV\_Nr» ist. (Der Name der Spalte in der Tabelle) Falls ja fragen wir noch, ob es leer ist oder ob es mit Daten gefüllt ist. Falls ja wird hier im Beispiel, das Property \_SRNr mit dem Inhalt gefüllt. Falls nein, geht der Kompilierer weiter.   
  
Das macht es nun mit jedem Item in der Tabelle. Es Fragt die verschiedenen IF-Statements ab. Bis es an dem ZER\_Verrechen IF-Statement ein True bekommt. Wenn dies der Fall ist, müssen wir nun unterscheiden, in welche Verrechnungsart, die Stunden hinter dem Rapport, welchen wir gerade bearbeiten, diese Stunden dazugehörig sind.

Der Rapport wurde beim erstellen deklariert, als was er im Moment im System abgespeichert werden soll. (Verrechnen, Warten, Garantie, Kulanz, nicht Verrechnen.) Diese werden dann mit auf der Z\_Erfassung Tabelle erfasst. Somit können wir dies unterscheiden. In der Spalte ZER\_Verrechnen werden für diese Verrechnungsarten Buchstaben hinterlegt. Damit können wir das nun unterscheiden. Mittels einer Select Case wird eine Abfrage gestaltet, welche auf diese Buchstaben abzielt:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Je nachdem, welcher Buchstabe angegeben wurde, wird es in dieses Property abgespeichert.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Da wir nun die Werte von einem ServiceRequest haben, können wir diese nun rechnen.

Ist Verrechenbar: Verrechnet + Warten (KundenSicht)

IST: \_Verrechnet + Warten + nichtVerrechnet, + Kulanz + Garantie (boSicht)

Diese Werte in den Properties, welche die gesamten zusammengefassten Werte eines ServiceRequests enthalten, werden nun als Objekt in der Liste Items in der Klasse PUCalc hinzugefügt.

Diesen Durchlauf wird nun so oft vollzogen, wie es ServiceRequests in der ProjektAnalyse gibt.   
  
Wir sind nun wieder zurück in der Klasse PUCalc. Hier gibt es nämlich noch 3 Weitere Methoden, welche gleich im Konstruktor aufgerufen werden. Sobald die Methode Calculate ihre Durchläufe und Berechnungen vollzogen hat, wird die Methode CalcTotal() verwendet.

###### CalcTotal

Diese Methode sorgt dafür, dass nun von jedem Rapport welcher in die Items hinzugefügt worden war, ein Total gerechnet wird in jeglichen Kategorien. Verrechnen, Warten, Kulanz, Garantie und nicht Verrechnen. Zusätzlich noch Total-Ist, alle Ist-Werte von allen Rapporten zusammengerechnet und alle Verrechenbar (KundenSicht) zusammengerechnet. Somit haben wir die Werte

* \_totalVerrechnet,
* \_totalWarten,
* \_totalKulanz,
* \_totalGarantie,
* \_totalNichtVerrechnet,
* \_totalIst
* \_Verrechenbar(KundenSicht)

Diese Werte können wir nun von aussen aufrufen und verwenden. In der Fill-Methode.

###### PBarProgressADRIST

Ist eine Methode, die den Prozentualen Wert als Property liefert.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dabei wird das totalIst \* 100 / totalSoll gerechnet. Zur Absicherung, falls ein Wert höher wie 100 herauskommt, ist noch eine IF-Abfrage notwendig. Denn die Progressbar in der ProjektAnalyse oder in der SSV nimmt keinen höheren Wert wie 100 entgegen.   
Der Prozent-Wert wird als Integer (Zeigt nur ganze Zahlen an) in das Property «\_ProzProgressADR» gespeichert, welcher von aussen aufgerufen werden kann.

###### PBarProgressSSV\_KS

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Das selbe wird in dem Prozentrechner für die ServiceRequests auch gemacht, jedoch wird nicht das ganze totalIst verwendet, sondern der Wert Verrechenbar(KundenSicht).

##### Darstellung

###### Grid

Solange ADR\_ID nicht 0 ist. Sonst wird das Befüllen übersprungen.

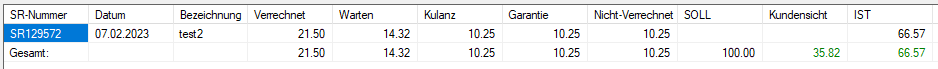
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die einzelnen Werte der Rapporte werden hier aus der Liste der PUCalc Klasse ausgelesen und in einer Row dargestellt im BoGridSR.

Sobald das letzte Item aus der Liste durchlaufen wurde, springt der Kompilierer aus der ForEach heraus und generiert noch die Letzte Row, welche alle Gesamt-Werte ausliest aus der Klasse PUCalc.  


Somit wurde eine Übersicht als boGrid dargestellt:



###### Progressbar

Für die Prozent-Anzeige (Progressbar) wird der Wert welcher aus der Klasse PUCalc gerechnet wurde mit der Methode «PBarProgressADRIST», verwendet.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Somit Zeigt die Progressbar die genaue Prozent an und auch das Label, welches die Prozent-Zahl anzeigt auch.

###### Farben von IST und KundenSicht

Um die Werte der IST und KundenSicht dynamisch zu machen, wie im Detaillierten Auftrag angegeben, müssen wir zuerst definieren, an welcher Position sich die Schrift-Farbe ändern soll.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Hier in diesem Abschnitt sind wir noch in der Klasse ctlADR, in welcher die Position definiert, wird im Grid. Sprich in der letzten Row, am 2. und 3. Letzten Platz. Spalten IST und KundenSicht.

Anhand der Methode GetColor(Proz as Integer) wird nun der Wert, welcher in den Einstellungen gewählt wurde, hier im Beispiel 80%, aus der Mandanten-Tabelle ausgelesen   
Der mitgelieferte Prozentwert, welcher im Konstruktor angegeben werden muss, wird mit den Einstellungen verglichen. Falls der Prozentsatz höher sein Soll wie die Einstellung, wird die Farbe Rot angezeigt im Grid. Falls er kleiner sein sollte, sogar kleiner als die Einstellungen -10 dann wird die Zahl Grün angezeigt. Jedoch wenn die Zahl kleiner als die Einstellungen ist aber grösser als die Einstellungen -10 dann wird die Zahl sich orange verfärben.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Buttons

Hier habe ich 5 Buttons oben Links ins Control hinzugefügt, diese werden dann später für die AddBeleg, AddSR, Delete, Refresh und Settings Befehle verwendet.   
Für die Images welche darauf angezeigt werden wird folgender Code Verwendet.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dabei wird im Code-behind die Images der Buttons angesteuert. Wir haben für solche Icons eine Bibliothek, welche auf dem oben angegebenen Pfad zu erreichen ist. Mit der Methode GetImage16 wird das angegebene Bild mit dem Namen auf ein Format 16x16 Pixel zurückgegeben(falls vorhanden) und somit auch angezeigt beim Initialisieren des Controls, da dieser Befehl im InitializeControl liegt.

### boSearchList

Auf dem Linken Panel, direkt unterhalb der Buttons wird die boSearchList (boSL) hinzugefügt werden, welche die Belege und auch die SR’s, welche mit den Buttons hinzugefügt wurden, anzeigt.  
um die boSL nun in das Panel zu adden wird es auch im Code-behind hinzugefügt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die boSL wurde als Klassen-Variabel deklarieren, jedoch instanziieren wird sie erst hier als New.   
Bei der Instanziierung wurde angegeben, dass die boSL «WithEvents» handelt. Daher können hier einige Angaben zusätzlich gemacht werden wie z.B. ContextMenuStrip welches ein Menu ist und beim Betätigen der rechten Maustaste ausgelöst wird.

Zusätzlich wird auch definiert auf welche Tabelle die boSL zugriffen soll. Hierbei wird die viewPA\_ProjAn verwendet. Welche uns wie im Abschnitt Datenbank erklärt wurde warum.

Mit den RowDBFiel\_Tile-Items werden die Untertitel hinzugefügt, welche im Auftrag gefordert wurden. Die Nr. wird ein bold-Text sein und grösser wie die Untertitel. Darunter werden dann die Untertitel Bezeichnung und Datum im Format «dd.MM.yyyy» angezeigt. Geordnet wird dabei nach der Nummer(SR und Belege).

Am Schluss wird die boSL dem Panel hinzugefügt mit dem Dockstyle Fill, das heisst, dass sich die boSL auf die Grösse vom Panel anpassen und einmitten wird.

### boGrid

Auf der rechten Seite des Controls ist ein weiteres Panel ersichtlich, welches für das boGrid genutzt wird auch hier ist das einfügen, definieren und Datenhinzufügen im Code-behind realisiert.

Das ist jedoch etwas grösser als die boSL. Dabei werden viele verschiedene Sachen definiert und benutzt:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Auch das boGrid wurde als Klassen Variable definiert und hier im InitializeControl, welches nach dem Konstruktor die 2 Methode ist die ausgeführt wird, initialisiert.  
Das boGrid wird mit einigen Optionen definiert, welche wichtig sind für die Darstellung der Daten später.   
Eine Fixe Reihe wird definiert mit Rows.Count = 1 und Rows.Fixed = 1. Diese wird als HeadRow verwendet, in welcher alle Captions geschrieben werden.

Um das boGrid schön zu gestalten, und auch genügend platz zu haben wird die letzte Column erweitert bis ans Panelende. Dafür wird eigens eine leere Column erstellt.

### Styles von boGrid

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
Bevor wir die einzelnen Columns erstellen, werden hier die Styles erstellt, welche verwendet werden sollten.

Text: Der Text wird vom Datentyp String sein und auf die Linke Seite im Tabellenkasten zentriert geschrieben

Dezimal: Die Ausgabe wird im Datentyp Dezimal sein und auf der rechten Seite im Tabellenkasten zentriert geschrieben. (Zahlenwerte im Format #,##0.00 z.B 1,000.50.-)

Date: Wird vom Datentyp Date sein und auch auf der linken Kastenseite zentriert geschrieben werden. Dabei wird es folgendes Format haben(dd.MM.yyyy)

Diese Definitionen werden als Items auf der boGrid gespeichert und können mit dem angegebenen String verwendet werden:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Nur als Beispiel, es sind zu viele Cols um alle aufzuzeigen jedoch hieran diesem Beispiel ist ersichtlich wie ich eine Column erstelle.   
Dabei wird sie definiert, geadded, den Style hinzugefügt und auch die Spaltenbreite definiert.   
Das Caption wird dann in der HeadRow (Rows.Count = 1) angezeigt.

Das wird nun mit allen Cols gemacht, welche wir sehen möchten.:

* SR-Nr. (Text)
* Datum (Date)
* Bezeichnung (Text)
* Verrechnet (Dezimal)
* Warten (Dezimal)
* Kulanz (Dezimal)
* Garantie (Dezimal)
* Nicht Verrechnet (Dezimal)
* IST (Dezimal)
* Und der Puffer, welcher leer ist um eine schöne Darstellung zu ermöglichen.

### Laden von RegForm

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

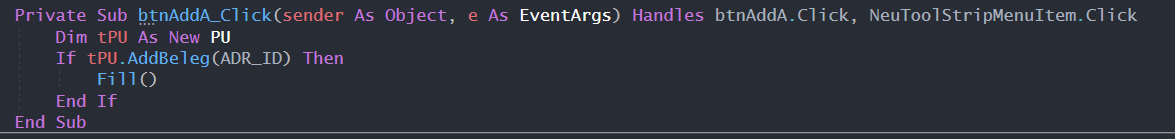
Mit diesem Code, werden die Spaltenbreiten welche der Benutzer vorgängig ein Mal verändert haben könnte, geladen. Dies wird geladen von der Registry unter dem Namen des Controls\boGrids\GridSR mit den entsprechenden Werten(Spaltenbreiten) jeder Column.

Beim Schliessen des Controls werden diese Spaltenbereite-Werte in der gleichen Weise in die Registry gespeichert. (Unterschied ist anstelle GetRegForm wird die Methode SaveRegForm verwendet.)

### Buttons und EventHandlers

In WindowsForms werden Events mitgeliefert, welche man abfangen kann. Diese verwenden wir sehr oft, um Buttons zu realisieren. Wie hier zum Beispiel.

### AddBeleg



Dies ist der AddA Button, welcher das Event «Click» auslöst. Wenn dieser Button gedrückt wird, löst er diese Programmierung aus, welche uns zu einer Methode führt, welche in einer anderen Klasse geschrieben wurde. Mit einer Gezielten Beziehung wird diese dann aufgerufen und ausgeführt.   
Mit dem Übergabeparameter im Konstruktor, welcher von der Methode verlangt wird, arbeitet diese dann.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Hier ist die Funktion welche ausgeführt wird. Sie ruft anhand der Adressen-ID einen Dialog auf, welcher alle Belge der Adresse zeigt, welche den Status «A» haben. «A» seht für Auftrag in unserem Auftragswesen.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Nach dem Auswählen aus dem Dialog, wird beim bestätigen mit «Übernehmen» einen SQL abgesetzt, welcher genau die Adresse verwendet und auch das ausgewählt Objekt.   
Gespeichert wird dies in die PA\_ProjAN Tabelle. Es wird jedoch vorher noch überprüft, ob es schon einen Eintrag mit dem selben Beleg schon enthält, falls ja, wird nichts bearbeitet.  
Der SQL – Insert ist ein einfach Insert in die Tabelle welcher durch unser API auf der Mandanten DB Abgesetzt wird.

### AddSR

Das gleiche wird auch für die SR’s gemacht. Das Abhandeln des Click-Events, mit der gezielten Beziehung auf die PU Klasse, in welcher dann eine Methode läuft welche einen Dialog für die SR’s von der Adresse öffnet. Wenn die Auswahl getroffen und bestätigt wurde, wird auch wieder ein Insertbefehl durch das boAPI auf die Mandanten-DB durchgeführt, welches zuerst schaut ob es den Datensatz schon in der Tabelle gibt. Falls nein wird er gespeichert sonst nicht.

### Delete Beleg/SR

## SSV-Control

# Kontrollieren

## Testprotokoll

## Testbericht

# Auswerten

## Reflexion

# Glossar

# Abbildungsverzeichnis

# Quellenverzeichnis

Auto Inkrement MS SQL

https://stackoverflow.com/questions/3359170/auto-increment-primary-key-in-sql-tables

# Anhang

## Anleitung

## MSSQL

## VB.Net