

SAP-Projekts



Umsetzung



Entwicklung einer SAP-Transaktion zum Generieren des internen Berechtigungsformular (Projekt für IHK-Abschlussprüfung)



/LWVH/RS_RCHNK Auswert (Rechnungen je nach Org-ID)



/LWVH/VITOS_AUFG (csv-Dateien auf dem SAP-Applikationsserver)



/LWVH/VITOS_BNKA (csv-Dateien auf dem SAP-Applikationsserver)



Code Verbesserung



/LWVH/ZFI_FAELLIGKEITEN_DEB1



/LWVH/ZFI_FAELLIGKEITEN_DEB2



Zusätzliche Projekte



Taschenrechner



Kalender



Römische Zahlen umrechnen



**Dokumentation zur betrieblichen Projektarbeit
IHK-Abschlussprüfung 2024**

**Entwicklung einer SAP-Transaktion
zum Generieren des internen Berechtigungsformular**

Prüfungsbewerber:

Sefa Alioglu

Schulstr 8

34277 Fuldabrück

Fachinformatik für Anwendungsentwicklung

Praktikumsbetrieb:

Landeswohlfahrtsverband Hessen

Ständeplatz 6-10

34117 Kassel



Inhaltsverzeichnis

<u>Dokumentation zur betrieblichen Projektarbeit</u>	i
<u>1 Einleitung</u>	1
<u>1.1 Projektbeschreibung</u>	1
<u>1.2 Projektanalyse</u>	1
<u>1.3 Konzeption Phase</u>	2
<u>1.4 Projektentwicklung und Implementierungsphase</u>	2
<u>1.5 Qualitätsmanagement</u>	2
<u>1.6 Abschlussfazit</u>	2
<u>2 Projektbeschreibung</u>	4
<u>2.1 Projektumfeld</u>	4
<u>2.2 Projektumsetzung</u>	4
<u>2.3 Projektziel</u>	4
<u>2.4 Projektanalyse</u>	5
<u>2.4.1 Ist-Analyse</u>	5
<u>2.4.2 Soll-Analyse</u>	5
<u>2.4.3 Projektbegründung</u>	5
<u>2.4.4 Lastenheft</u>	6
<u>3 Konzeption Phase</u>	7
<u>3.1 Projektphasen</u>	7
<u>3.2 Ressourcenplanung</u>	7
<u>3.2.1 Hardware</u>	7
<u>3.2.2 Software</u>	7
<u>3.3 Zielplattform</u>	9
<u>3.4 Ablaufplan für papierbasierten Prozess</u>	10
<u>3.5 Architekturdesign</u>	10
<u>3.6 Pflichtenheft</u>	11
<u>3.7 Anwendungsfälle (Use Case)</u>	12
<u>3.7.1 Akteure</u>	12
<u>3.7.2 Funktionalitäten</u>	12
<u>3.7.3 Assoziationen und Erweiterungen</u>	12
<u>3.7.4 Erweiterte Optionen</u>	13
<u>3.8 Datenmodell</u>	13
<u>3.9 Erstellung einer Excel-Tabelle für SAP-Daten</u>	13
<u>3.10 Entwurf der Benutzer Oberfläche (GUI)</u>	16
<u>3.10.1 Eingabe Feld für Benutzende.</u>	16
<u>3.10.2 Funktionalität der Buttons:</u>	16
<u>3.10.3 Ausgabefeld für die Stammdaten des Benutzenden</u>	16
<u>3.10.4 Konzept zur Sichtbarkeit der Felder und Buttons:</u>	17
<u>4 Projektentwicklung und Implementierungsphase</u>	19

<u>4.1</u>	<u>Transportieren Daten in SAP</u>	19
<u>4.2</u>	<u>Erstellen eine Tabelle zum Verknüpfen von Rollen</u>	19
<u>4.3</u>	<u>Erstellen GUI</u>	19
<u>4.4</u>	<u>Entwicklung der Methoden</u>	20
<u>4.4.1</u>	<u>Datenbankverarbeitung und SQL-Abfragen</u>	20
<u>4.4.2</u>	<u>Ablesen GUI und Anzeigen auf der GUI</u>	21
<u>4.4.3</u>	<u>Ausgewählte Daten anzeigen</u>	22
<u>4.4.4</u>	<u>Smart Forms</u>	22
<u>5</u>	<u>Qualitätsmanagement</u>	25
<u>5.1</u>	<u>Testphase</u>	25
<u>5.1.1</u>	<u>Unit-Tests</u>	25
<u>5.1.2</u>	<u>Integrationstests</u>	25
<u>5.1.3</u>	<u>Systemtests</u>	25
<u>5.1.3.1</u>	<u>White-Box-Tests:</u>	25
<u>5.1.3.2</u>	<u>Black-Box-Tests:</u>	25
<u>5.2</u>	<u>Fehleranalyse</u>	26
<u>6</u>	<u>Abschlussfazit</u>	27
<u>6.1</u>	<u>Fachliches und Persönliches Fazit</u>	27
<u>6.2</u>	<u>Lessons-Learned</u>	27
<u>6.3</u>	<u>Projektübergabe</u>	28
<u>7</u>	<u>Literaturverzeichnis</u>	29
<u>8</u>	<u>Glossar</u>	30
<u>9</u>	<u>Anhänge</u>	32
<u>9.1</u>	<u>Lastenheft</u>	32
<u>9.1.1</u>	<u>Verarbeitung der Daten</u>	32
<u>9.1.2</u>	<u>Darstellung der Daten</u>	32
<u>9.1.3</u>	<u>Sonstige Anforderungen</u>	32
<u>9.1.4</u>	<u>Zusammenfassung</u>	32
<u>9.2</u>	<u>Pflichtenheft</u>	33
<u>9.2.1</u>	<u>Plattform</u>	33
<u>9.2.2</u>	<u>Datenbank</u>	33
<u>9.2.3</u>	<u>Oberfläche</u>	33
<u>9.2.4</u>	<u>Geschäftslogik</u>	33
<u>9.3</u>	<u>Datenbankmodell</u>	34
<u>9.4</u>	<u>Ein Teil des Antrags</u>	35
<u>9.5</u>	<u>Ablaufplan für papierbasierten Prozess</u>	36
<u>9.5.1</u>	<u>IST-Stand</u>	36
<u>9.5.2</u>	<u>Ausführlicher Ablaufplan</u>	37
<u>9.6</u>	<u>Use-Case Diagramm</u>	38
<u>9.6.1</u>	<u>Use-Case Diagramm fürs Projekt</u>	38
<u>9.6.2</u>	<u>Erweiterte Use-Case-Diagramm</u>	39
<u>9.7</u>	<u>Projektphasen mit Zeitplanung in Stunden</u>	40
<u>9.8</u>	<u>MVC-Sequenzdiagramm</u>	41

<u>9.9</u>	<u>Entity-Relationship-Modell (ERM)</u>	41
<u>9.10</u>	<u>Quellcode</u>	42
<u>9.10.1</u>	<u>TOP – PROGRAM /LWVH/BERECHTIGUNGFORMULA.</u>	40
<u>9.10.2</u>	<u>P01 – CLASS LCL_BERECHTIGUNGFORMULA IMPLEMENTATION.</u>	42
<u>9.10.3</u>	<u>PBO - /LWVH/BERECHTIGUNGFORMULA_001.</u>	47
<u>9.10.4</u>	<u>PAI - /LWVH/BERECHTIGUNGFORMULA_I01.</u>	49

Einleitung

Im Rahmen meines Praktikums zum Fachinformatiker Anwendungsentwicklung beim Landeswohlfahrtsverband Hessen, nachfolgend LWV Hessen genannt, habe ich mein IHK-Abschlussprojekt durchgeführt. Das IHK-Abschlussprojekt ist nachfolgend „Projekt“ genannt.

Das Ziel des Projekts ist es, mit einer SAP-Transaktion¹, ein internes Berechtigungsformular zu generieren.

Das Lastenheft beschreibt die Anforderungen und Spezifikationen für die Entwicklung eines automatisierten Genehmigungsprozesses in SAP. Andere Themen des Lastenheftes, wie die Ist-Analyse, Soll-Analyse, werden in der Projektanalyse, Kapitel 3, behandelt.

Projektbeschreibung

Das Projekt zielt darauf ab, die manuelle Verarbeitung des papiergebundenen Berechtigungsformulars zu digitalisieren und digital in das SAP-System zu integrieren.

Mit Hilfe des Projekts werden auch Probleme mit alten Versionen des Formulars sowie unvollständig ausgefüllten Pflichtfeldern und Personalinformationen gelöst. Die Digitalisierung des Formulars wird sicherstellen, dass stets die aktuellste Version verwendet wird und verhindert Fehler bei fehlenden Eingaben.

Das Programm wird es die Mitarbeitende ermöglichen, ihre Anträge direkt in SAP auszufüllen und mithilfe von Smart Forms² in ein druckbares Format umzuwandeln.

Projektanalyse

Die Projektanalyse umfasst die Ist-Analyse, Soll-Analyse, Projektbegründung und Lastenheft. Es wurde festgestellt, dass die aktuellen papierbasierten Genehmigungsprozesse zeitaufwändig und fehleranfällig sind. Die anderen Themen eines Lastenheftes wurden als Anlage hinzugefügt. Die Einführung eines automatisierten Genehmigungsprozesses in SAP soll die Effizienz steigern und Fehler reduzieren. Damit wird eine klare Nachverfolgung des Berechtigungsantrags ermöglicht.

Auf Basis der Analyseergebnisse habe ich ein umfangreiches Konzept für die neue Software erstellt.

¹ Eine Transaktion ist eine Folge von SQL-Anweisungen, die eine Gruppe von miteinander verbundenen sind, die als Einheit behandelt.

² SAP Smart Forms bieten eine grafische Oberfläche mittels einem Form Builder in SAP, um eine Logik und das Layout eines Dokuments oder Formulars zu erstellen.

Hierfür ist Projektphasen mit Zeitplanung in Stunden wie im Folgende vorgesehen.

Phase	Zeit in Stunden
Projektanalyse	5
Konzeption Phase	25
Projektentwicklung und Implementierungsphase	32
Qualitätsmanagement	6
Erstellen der Dokumentationen	9
Projektübergabe	3
Summe	80

Tabelle 1.1: Projektphasen mit Zeitplanung in Stunden

Konzeption Phase

Die Konzeptionsphase umfasst die Erstellung des GUI-Interfaces und die Planung der Ressourcen. Ein Benutzende freundliches GUI-Interface wird entwickelt, das eine einfache Bedienung und Navigation anbieten soll. Das Pflichtenheft, das Architekturdesign und der Ablaufplan für papierbasierten Prozess befinden sich auch in diesem Abschnitt.

Projektentwicklung und Implementierungsphase

In dieser Phase habe ich das Implementieren in die Tat umgesetzt und die Anwendung in der von SAP ausgelieferten proprietären Programmiersprache ABAP entwickelt. ABAP (Advanced Business Application Programming) ist eine Programmiersprache, die speziell für SAP-Anwendungen entwickelt wurde.

Qualitätsmanagement

Nach Abschluss der Implementierung wurde die Anwendung ausgiebig getestet, um mögliche Fehler oder Unstimmigkeiten zu identifizieren und zu beheben. Das Qualitätsmanagement wurde in zwei Teile unterteilt. Sie sind Testphase und Fehleranalyse.

Abschlussfazit

Das Abschlussfazit wurde in drei Abschnitte unterteilt. Der Erste ist das fachliche und persönliches Fazit, das zweite ist Lesson-Learned sowie das dritte ist die Projektübergabe.

Die Lessons-Learned ist entscheidend, um Erkenntnisse und Verbesserungsmöglichkeiten aus dem Projekt zu ziehen. Herausforderungen, Erfolge und Bereiche wurden genannt, die verbessert werden sollen. Diese Erkenntnisse können für zukünftige Projekte sehr wichtig sein.

Bei der Projektübergabe wurden alle Anforderungen erledigt oder offene Punkte identifiziert und in einer Liste festgehalten, die weiterbearbeitet wird, um das Projekt vollständig abzuschließen.

Mit dem IHK-Abschlussprojekt konnte eine moderne Anwendung für den LWV Hessen entwickelt werden, welche einen deutlichen Mehrwert für das Unternehmen bietet. Die neuen Prozesse sind effizienter gestaltet und ermöglichen eine verbesserte Verwaltung sowie eine bessere Unterstützung der Zielgruppe.

Abschließend möchte ich mich bei meinem Ausbildungsbetrieb LWV Hessen für diese spannende Aufgabe bedanken sowie bei meinem Ausbilder für seine Unterstützung während des gesamten Projekts.

Projektbeschreibung

Projektumfeld

Der LWV Hessen ist ein hessenweiter Kommunalverband, der soziale Leistungen für behinderte, psychisch kranke sowie sozial benachteiligte Menschen finanziert und sie in ihrem Alltag und im Beruf unterstützt.

Der LWV finanziert Unterstützungsleistungen für Menschen mit Behinderungen, um ihnen ein selbstständiges Leben zu ermöglichen.

Der LWV gewährt finanzielle Unterstützung für Menschen, die aufgrund sozialer Benachteiligung oder besonderer Lebensumstände nicht in der Lage sind, ihren Lebensunterhalt selbständig zu bestreiten.

Projektumsetzung

Die Projektumsetzung fand ausschließlich in den Räumlichkeiten bei LWV Kassel statt.

Projektziel

Das Projekt zielt darauf ab, einen automatisierten Genehmigungsprozess für papierbasierte Formulare in SAP zu implementieren. Durch die Entwicklung eines maßgeschneiderten Programms wird es ermöglicht, dass Mitarbeitende ihre Berechtigungsanträge direkt in SAP ausfüllen können. Das Programm soll eine Berechtigungsformel enthalten, um sicherzustellen, dass nur autorisierte Mitarbeiter mit den entsprechenden Rollen³ die Anträge stellen und genehmigen dürfen.

Zusätzlich wird das Programm vorgesehen, dass es die ausgefüllten Informationen mithilfe von Smart Forms in ein druckbares Format umgewandelt werden kann. Nachdem das Formular gedruckt wurde, wird es durch die Verwaltung an die entsprechende Abteilung weitergeleitet. Diese Vorgehensweise ist aus internen Gründen erforderlich.

Die Automatisierung des manuellen Prozesses der Formularausfüllung einer Internberechtigung durch die Implementierung einer Lösung in SAP, um die Effizienz und Genauigkeit der Dateneingabe zu verbessern. Durch die Nutzung von soll es die Mitarbeitende ermöglicht werden, das Berechtigungsformular direkt in SAP auszufüllen.

³ Rolle ist die Tätigkeit einer Benutzerin und dient der Anzeige des Benutzerspezifischen Menüs im SAP. Es gibt für diesen Fall vordefinierte Rollen, die auch die Berechtigungen enthalten, mit denen in SAP arbeiten dürfen.

Projektanalyse

In diesem Abschnitt wurde das Projekt analysiert. Dafür wurde es in fünf Themen unterteilt: die Ist-Analyse, Soll-Analyse, Projektbegründung, Lastenheft.

Ist-Analyse

Die aktuellen papierbasierten Genehmigungsprozesse sind manuell und erfordern, dass Mitarbeiter ein physisches Formular ausfüllen, welches dann von verschiedenen Ebenen der Genehmigung durchlaufen. Dieser Prozess ist zeitaufwändig und anfällig für Fehler. Zum Beispiel nicht ausgewählte Berechtigungen, Fehler beim Ausfüllen von Pflichtfeldern oder unpassende Angaben. Die Dateneingabe erfolgt manuell, was zu Verzögerungen und möglichen Widersprüche führen kann.

Soll-Analyse

Das Ziel ist es, einen automatisierten Genehmigungsprozess in SAP zu implementieren. Mitarbeiter können ihre Anträge direkt in SAP auszufüllen und einreichen. Das Programm wird spezielle Berechtigungsniveaus enthalten, um sicherzustellen, dass nur autorisierte Personen die entsprechenden Rollen genehmigen dürfen. Zusätzlich wird das Programm die ausgefüllten Informationen mithilfe von Smart Forms in ein druckbares Format umwandeln können.

Auf diese Weise werden mögliche Fehler vermieden und der Prozess wird digitalisiert.

Mittels der Automatisierung des Prozesses soll die Dateneingabe verbessern sowie eine klare Nachverfolgung des Genehmigungsstatus ermöglichen.

Projektbegründung

Die Erstellung eines Antrags der benötigten Berechtigungen in SAP kann im Zweifel zu konkreten Problemen führen, da das Formular manuell ausgefüllt und manuell in SAP übertragen werden muss. Eine Korrektur von Fehlern erfordert ein hohes Maß an Aufwand. Hinzu kommt der Prozess bei der Freischaltung der Berechtigungen auch der hohe zeitliche Aufwand.

Auf der anderen Seite sind alle Daten schon in der SAP-Datenbank vorhanden, trotzdem müssen die Berechtigungen manuell übertragen werden.

Ein weiteres Problem ist das Fehlen einer Korrektheit im aktuellen Prozess. In diesem Fall gibt es zwei mögliche Fehler:

1. Nicht angekreuzte Berechtigungen, die unbedingt benötigt werden, um bei der Arbeit die Aufgaben zu erledigen.

2. Vergessene Unterschriften oder fehlende persönliche Daten.

Lastenheft

Bereits beschrieben wurde die Projekteinführung, die Projektbeschreibung, die Beschreibung der Ist-Analyse und der Soll-Analyse als ein Teil des Lastenheftes. Daher werden die zusätzlichen Anforderungen des Projektes in diesem Abschnitt aufgeführt.

Am Ende der Projektanalysephase wurde ein Lastenheft erstellt. Damit wird die Umsetzung des Programmes erleichtert. Dafür wurden die wichtigsten Anforderungen des Projektes vorgeschlagen und festgelegt.

Die Formulierung wurde mit Hilfe der Must-have, Nice-to-have und nicht relevant umgesetzt. Das heißt, dass in jeder Anforderung die Wichtigkeit aufgeschlüsselt nach „muss“, „würde gerne“ und „irrelevant“ festgehalten wird. Das vollständige [Lastenheft](#) befindet sich im Anhang 9.1 auf Seite 30.

Konzeption Phase

Die Konzeptionsphase umfasst die Erstellung des GUI-Interfaces und die Planung der Ressourcen. Es befinden sich auch Zielplattform⁴, Ablaufplan für den papiergebundenen Prozess, Architekturdesign, Pflichtenheft, Datenmodell, Anwendungsfälle (Use Case) und Erstellen Strukturen hier.

Projektphasen

Eine detaillierte [Zeitplanung](#) ist im Anhang 9.7 auf Seite 38. einzusehen.

Ressourcenplanung

Alle Ressourcen sind in den folgenden Kapiteln aufgelistet.

Hardware

🔧 Büroarbeitsplatz mit Desktop Rechner.

Software

🔧 Entwicklungsumgebung: SAP R/3 7.50.

🔧 Application Server: SAP NetWeaver for ABAP 7.52.

🔧 Datenbanksystem: HANA DB.

🔧 Betriebssystem: Windows 10.

SAP steht für „Systemanalyse Programmentwicklung“. SAP ist eine Kurzform davon. SAP-Software wird für die Verwaltung und Management der Geschäftsprozessen wie Finanzen, Personalwesen, Produktion, Vertrieb und Logistik eingesetzt. SAP bietet nicht nur Verarbeitung der Daten und Informationen und auch Reduzierung der Arbeitsbelastung, zu erleichtern und verbessern.⁵

🔧 *Was ist ERP?*

„ERP steht für Enterprise Resource Planning. Dabei geht es um die zeit- und bedarfsgerechte Planung aller Ressourcen im Unternehmen (wie z. B. Kapital, Personal oder Betriebsmittel). Mit einer ERP-Software lassen sich diese Unternehmensprozesse und -ressourcen automatisieren, planen und steuern.“

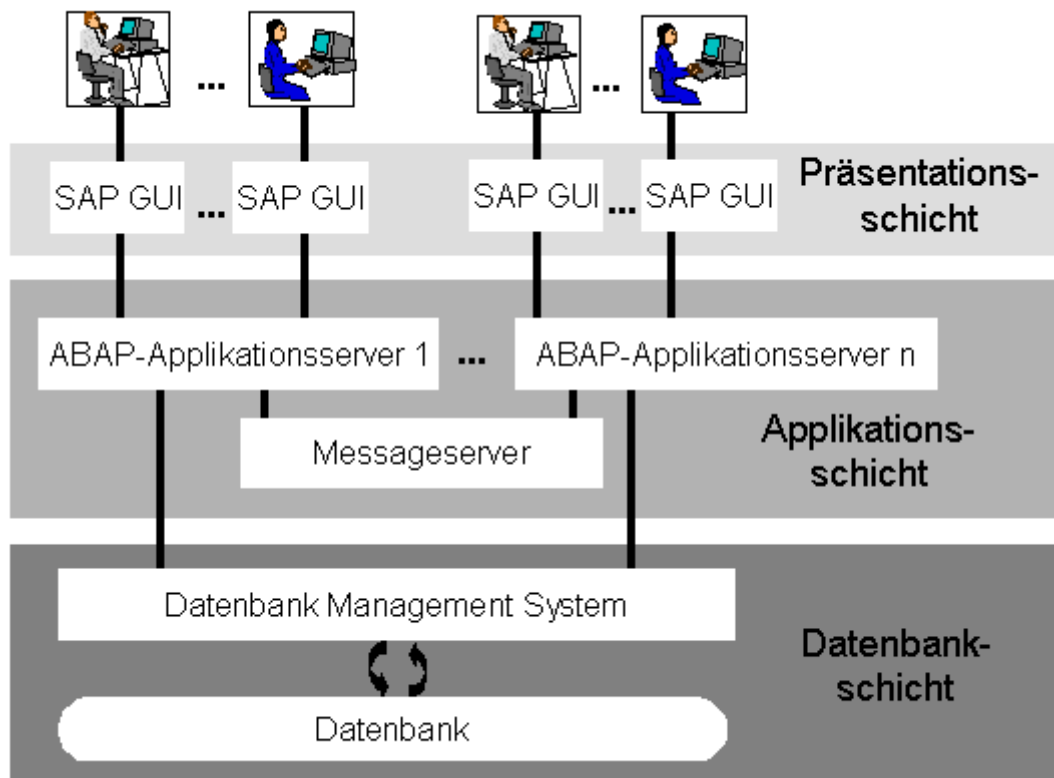
⁴ Zielplattform ist Abschnitt 3.3 bereitgestellt.

⁵ (<https://www.sap.com/germany/about/what-is-sap.html>, 2024)

⚙️ Was ist SAP R/3?

Die drei Systeme folgen zeitlich aufeinander, wobei SAP R/3 das älteste der drei SAP-ERP-Systeme ist. SAP R/3 kam 1992 auf den Markt. Das „R“ im Produktnamen stand für „Real-time data processing“ und die „3“ für den dreistufigen Aufbau des Systems aus Datenbank, Applikationsserver und Benutzer Oberfläche.“⁶

⚙️ Übersicht über den NetWeaver AS ABAP



Software-technische Sicht⁷

Durch die Verteilung des SAP-Systems auf drei Ebenen wird die Systemlast verteilt. Dies führt zu einer besseren Systemleistung. Die Architektur des SAP-Systems ermöglicht es, die Anwendungsschicht und die Datenbankschicht auf separaten Hosts zu installieren. Ebenso können diese so über das Netzwerk kommunizieren. Dies reduziert die Belastung des

⁶ (<https://www.gambit.de/wiki/unterschiede-sap-r3-ecc-und-s4hana/>, 2024)/

⁷

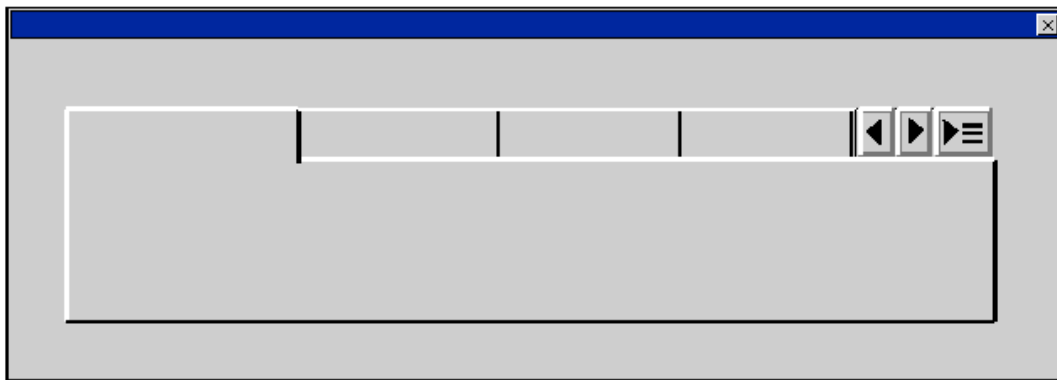
(https://help.sap.com/saphelp_autoid2007/helpdata/de/fc/eb2e97358411d1829f0000e829fbfe/content.htm?no_cache=true, 2024)

Datenbanksystems, das den zentralen Datenspeicher für SAP NetWeaver AS ABAP enthält, erheblich.⁸

Was ist Dynpro und Tabstrip und Tab?

Die Kernfunktion von Dynpros besteht in der Bereitstellung einer Benutzerfreundlichen Schnittstelle, welche die Interaktion zwischen dem Benutzende und dem SAP-System optimiert.

Ein TabStrip oder TabStrip-Control stellt ein Bildschirmobjekt dar, welches aus mehreren Seiten besteht, die auch Tab genannt werden.⁹



Zielformat

Wie bereits unter 1.1 (Projektbeschreibung) erwähnt, soll das Abschlussprojekt als eigenständiges Programm entwickelt werden.

Die Zielformat für das Abschlussprojekt ist ein internes Programm, das in ABAP entwickelt wird. Durch die Verwendung von ABAP zur Implementierung wird sichergestellt, dass Mitarbeitende ihre Berechtigungsanträge direkt in SAP ausfüllen können.

Das Release und die Aktualisierung auf eine neue Version der Anwendung werden vereinfacht sein. Der Einsatz von ABAP als Programmiersprache sowie der HANA-Datenbank ermöglicht es, dieses Projekt erfolgreich umzusetzen.

⁸ (<https://rz10.de/knowhow/sap-netweaver-application-server-abap-as-abap/>, 2024)

⁹ Grafisch: (https://help.sap.com/doc/saphelp_gbt10/1.0/de-DE/4a/44b861954c0453e10000000a421937/content.htm?no_cache=true, 2024)

Ablaufplan für papierbasierten Prozess

Als erstes erstellt ein Benutzende, der Berechtigungen in SAP erhalten soll, einen Antrag. Dazu werden die erforderlichen Berechtigungen auf einem physischen Dokument ausgefüllt. Danach wird dieser Antrag vom Vorgesetzten geprüft, sofern keine Fehler vorliegen, wird er genehmigt. Der genehmigte Antrag wird dem zuständigen Fachbereich (Fachbereich Datenverarbeitung) zur weiteren Genehmigung vorgelegt. Der Fachbereich prüft die Korrektheit und Vollständigkeit des Antrags. Wenn alles in Ordnung ist, werden die angeforderten Berechtigungen freigeschaltet. Die Verbandshauptkasse, im Folgenden VHK genannt, erhält das Formular zur Kenntnis, sobald ein Kreuz bei papiergebundenen Freigabesystem und unbegrenztem Erfassen/Anordnen gesetzt ist, um Unterschriften zu prüfen, wenn eine Rechnung von dem entsprechenden Mitarbeitende erstellt werden. Der Ablaufplan wurde in zwei Teile dargestellt. Einmal der IST-Stand und einmal ausführlich, um die Vorgehensweise des Antrages zu verstehen. [Ein Teil des Antrags](#) ist auch als Anhang auf Seite 33 beigefügt. Der [Ablaufplan](#) für papierbasierten Prozess wurde im Anhang 9.5 auf Seite 34 bereitgestellt.

Architekturdesign

Als Architekturdesign wurde Das Model-View-Controller (MVC) verwendet.¹⁰

Der View behandelt die grafischen und textbasierte Ausgaben auf dem GUI. Der View stellt Eingangs- und Ausgangsdaten zur Verfügung. Dies sind Menüs, Dialogfenster, Drucktasten, Smart Forms usw. Die Visualisierung wird über View aufgeführt. In diesem Projekt bedeutet es [PBO](#).¹¹

Der Controller interpretiert und überwacht die Eingabedaten des Benutzenden. Die Eingabedaten werden auf Model und View zugeführt, um die Änderungen des Benutzenden auszuwerten und darauf zu reagieren. Der Controller besteht beispielsweise aus Form, Methode, Funktion usw. aus. In SAP nennt sich das [PAI](#).¹²

[Das Model](#) dient als Objekt der Verwaltung der Eingabedaten des Benutzenden. Dies bedeutet, es ist verantwortlich für die Datenverarbeitung. Im Übrigen läuft es im Hintergrund und koordiniert die Interaktionen mit PBO und PAI. Für diesen Fall wurden ABAP-Klassen und -Funktionsbausteine verwendet, um die Anwendungslogik zu implementieren. Dies

¹⁰ (MVC Design Pattern | SAP Help Portal, 2024)

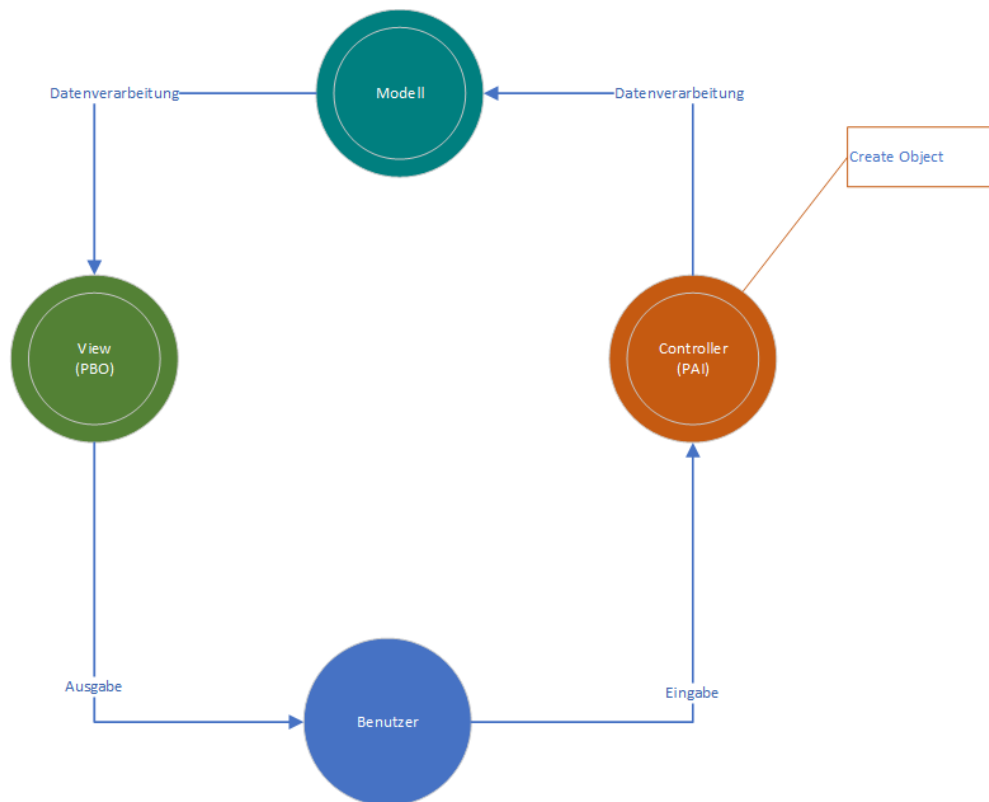
¹¹ PBO, Process Before Output, wird von der ABAP-Laufzeitumgebung vor dem Erscheinen Bildschirm an die Präsentationsschicht ausgelöst.

¹² PAI, Process After Input, wird ausgelöst bei einer Aktion vom User auf dem Bildschirm

können beispielsweise Zugriffe, Lesen, Manipulation und Speichern der Daten sein. Der Prozess steht in Verbindung mit der HANA-Datenbank

MVC erleichtert die Weiter- und Entwicklung, Wartung, Erweiterung und Debugging des Projektes.

Die obigen Ausführungen stellen im Folgende als Grafisch dar und [MVC-Sequenzdiagramm](#) befindet sich Anhang 9.8 und auf Seite 39.



MVC Design Pattern¹³

Pflichtenheft

Das Pflichtenheft wurde während der Konzeption-Phase erstellt. Dieses Dokument dient als Leitfaden für die konkrete Umsetzung der im Lastenheft der Anforderungen. Das Pflichtenheft ist das Ergebnis der Planung und Konzeptionierung, die darauf abzielt, einen automatisierten Genehmigungsprozess in SAP zu realisieren. [Das Pflichtenheft](#) befindet sich Anhang 9.2 und auf Seite 31.

¹³ (Model-view-controller - Wikipedia, 2024)

Anwendungsfälle (Use Case)

Das Use-Case-Diagramm zeigt die Interaktionen zwischen den Akteuren (Mitarbeitende, Berechtigungsadministrator und System).

Akteure

- Benutzende, Mitarbeitende, die das System verwendet, um einen Antrag zu stellen.
- Berechtigungsadmin: Eine spezielle Art von Benutzende mit erweiterten Rechten zur Verwaltung von Berechtigungsanträgen.

Das System stellt das Programm mit dem die Benutzende und der Berechtigungsadmin arbeiten zur Verfügung.

Funktionalitäten

- Anzeigen: Benutzende können bestehende Berechtigungen anzeigen lassen.
- Anlegen: Benutzende können neue Berechtigungen erstellen.
- Ändern: Benutzende können bestehende Berechtigungen ändern.
- Anlegen mit Vorlage: Benutzende können neue Berechtigungen basierend auf einer vorhandenen Vorlage erstellen.
- Löschen: Eine spezielle Funktion, die nur für Berechtigungsadmin verfügbar ist, der Berechtigungen löschen darf.

Assoziationen und Erweiterungen

- Berechtigungsadmin: Eine Erweiterung des Benutzende mit zusätzlichen Berechtigungen.
- Speichern: Eine Funktion zum Speichern von Änderungen und neuen Einträgen.
- Drucken: Eine Funktion zum Drucken des bereits bestehenden oder gestellten Antrags zu drucken.

Der Benutzende interagiert mit dem System, um Berechtigungsanträge zu verwalten oder einen Antrag zu stellen.

Das System speichert Änderungen und neue Einträge gemäß den Anforderungen.

Je nach Option werden Stammdaten des eingegebenen Benutzenden aufgerufen. [Die Optionen](#) sind bereits Abschnitt 3.10.2 dargestellt.

[Das Diagramm](#) wurde im Anhang auf 9.6 auf Seite 36 bereitgestellt.

Erweiterte Optionen

Es gibt eine Assoziation zum Admin, gewünschte Änderungen/Berechtigungen anzeigen zu lassen.

Diese Änderungen/Berechtigungen können auch spezifische Berechtigungen aufrufen, die bereits im System vorhanden sind.

Ein spezielles Programm wird aufgerufen, um die gewünschten Berechtigungen freizuschalten.

Diese Aktion erfordert ebenfalls eine Assoziation zum Admin um die Freischaltung zu ermöglichen.

Das Programm, das die Freischaltung der gewünschten Berechtigungen ermöglicht, wird in den Prozess einbezogen.

Es ruft die gewünschten Berechtigungen auf, um sicherzustellen, dass die richtigen Änderungen vorgenommen werden.

Das erweiterte Diagramm wurde im [Anhang auf 9.6.2](#) bereitgestellt.

Datenmodell

Das Datenmodell besteht aus den Entitäten /lwwh/bf_dyn_agr, agr_define, DD02L, DD03L und Strukturen, die im Abschnitt 4.1 dargestellt sind. Die Entität agr_define stellt autorisierten Rollen zur Verfügung. Die Entität /lwwh/bf_dyn_agr wurde als Datenbanktabelle in SAP erstellt, um Rollen zu verknüpfen. Tabname der Tabelle /lwwh/bf_dyn_agr wird in der Tabelle DD02L und die Feldnamen der Tabelle /lwwh/bf_dyn_agr in Tabelle DD03L gespeichert. Diese Beziehungen sind im Anhang 9.3 auf Seite 32 als [Datenbankmodell](#) dargestellt.

Erstellung einer Excel-Tabelle für SAP-Daten

Dieser Prozess beschreibt die Schritte zur Erstellung einer Excel-Tabelle, um Daten zu verwalten und zu analysieren. Die Excel-Tabelle dient als praktisches Werkzeug zur Organisation und Analyse von Daten, die im SAP-Systemen importiert wurden.

Feldbezeichner												
Rolle	usr.	Komponente	Vollständiger Name	Komponententyp	Strukturname	Domäne	Datenobj.	Kurzbeschreibung	kurz (10)	mittel (15)	lang (20)	Überschrift (40)
Support		support	sons_support	/LWVHLsons_support	/LWVHBF_S_DYNP_SONS	/LWVHIBA_D_CHKEX	XFELD	Support	Support	Support	Support	Support
Revision		revision	sons_revision	/LWVHLsons_revision				Revision	Revision	Revision	Revision	Revision
Hotline		hotline	sons_hotline	/LWVHLsons_hotline				Hotline	Hotline	Hotline	Hotline	Hotline

Zuerst wurden die Anforderungen für die Excel-Tabelle analysiert, einschließlich der benötigten Datenfelder und Struktur.

Basierend auf den Anforderungen wurde ein Entwurf für die Excel-Tabelle erstellt, der die Spaltenüberschriften und Datenfelder definierte.

Abschnitt: Diese Spalte kennzeichnet den Abschnitt des Berechtigungsformulars, zu dem die Daten gehören. Jede Rolle kann mehrere Abschnitte haben, und jeder Abschnitt wird in einer eigenen Zeile aufgeführt.

Überblick
2.2 Berechtigung für Buchungskreise
2.3 a) Kassenwesen
2.3 b) Haushaltswesen
2.3 c) Rechnungswesen
2.3 d) Sonstiges
2.3 e) Nur Beauskunftung
2.4 a) Verfügungsberechtigungen für Sachkonten
2.4 b) Verfügungsberechtigungen für Kostenstellengruppen, Profitcenter
2.4 c) Verfügungsberechtigungen für Geschäftspartner
2.4 d) Anordnen
2.4 Verfügungsberechtigungen für Innenauftragsnummern
3. Faktura und 4. Verwahrgeless

Mit den Anforderungen und dem Entwurf als Leitfaden wurde die Excel-Tabelle erstellt.

Die Spaltenüberschriften wurden entsprechend den definierten Datenfeldern formatiert und benannt.

Die Tabelle wurde so angeordnet, dass sie eine effiziente Datenverwaltung und -analyse ermöglicht. Dabei wurden folgende Spalten wie im Folgende definiert.

Rolle: Diese Spalte enthält den Namen der Rolle, für die die Informationen gesammelt wurden. Jede Rolle wird in einer eigenen Zeile aufgeführt.

Komponentenname: Diese Spalte gibt den Namen der Komponente oder des Attributs innerhalb der SAP-Struktur an. Es handelt sich um den spezifischen Namen der enthaltenen Elemente.

Vollständiger Name: Diese Spalte gibt den vollständigen Namen der Komponente oder des Attributs an. Der vollständige Name wird durch die Kombination des Kurznamens und des Komponentennamens gebildet.

Komponententyp: Hier wird der Typ der SAP-Komponente angegeben, zu der das Attribut gehört.

Strukturname: Diese Spalte gibt den Strukturnamen in SAP an, der mit der Komponente oder dem Attribut verbunden ist. Dies ist der Name der Struktur, in der die Daten in SAP gespeichert sind.

Strukturname	/LWVH/BF_S_DYNP_USR	Überblick
	/LWVH/BF_S_DYNP_BUK	2.2 Berechtigung für Buchungskreise
	/LWVH/BF_S_DYNP_VHK	2.3 a) Kassenwesen
	/LWVH/BF_S_DYNP_HW	2.3 b) Haushaltswesen
	/LWVH/BF_S_DYNP_RCHN	2.3 c) Rechnungswesen
	/LWVH/BF_S_DYNP_SONS	2.3 d) Sonstiges
	/LWVH/BF_S_DYNP_INFO	2.3 e) Nur Beauskunftung
	/LWVH/BF_S_DYNP_SAKO	2.4 a) Verfügungsberechtigungen für Sachkonten
	/LWVH/BF_S_DYNP_KSTL	2.4 b) Verfügungsberechtigungen für Kostenstellengruppen, Profitcenter
	/LWVH/BF_S_DYNP_GP	2.4 c) Verfügungsberechtigungen für Geschäftspartner
	/LWVH/BF_S_DYNP_ANORD	2.4 d) Anordnen
	/LWVH/BF_S_DYNP_INNENAUFTRAG	2.4 Verfügungsberechtigungen für Innenauftragsnummern
	/LWVH/BF_S_DYNP_FAKTURA_VWG	3. Faktura und 4. Verwahrgelass

Feldbezeichner: Hier wird der Feldbezeichner für jede Komponente oder jedes Attribut angegeben. Der Kurzname dient als eindeutige Bezeichnung für die Komponente. Dazu wurde kurz, mittel, lang und Überschrift deklariert.

Domäne¹⁴: Hier wird die Domäne angegeben, zu der das Attribut gehört. Eine Domäne definiert den möglichen Wertebereich eines Datenobjekts in Struktur.

Datentyp: Diese Spalte gibt den Datentyp an, der für das Attribut in der SAP-Domäne definiert ist. Der Datentyp legt fest, welche Art von Daten in einem Attribut gespeichert werden können, z. B. Char oder „d“ für Datum usw.

Kurzbeschreibung (SAP GUI): Hier wird eine kurze Beschreibung des Attributs oder der Komponente angegeben, wie sie im SAP GUI angezeigt wird. Diese Beschreibung soll die Benutzende helfen, das Attribut oder die Komponente zu identifizieren und zu verstehen.

Die erforderlichen Daten wurden aus dem papiergebundenen Berechtigungsantrag manuell in die Excel-Tabelle eingetragen.

Dabei wurden die Daten entsprechend den definierten Datenfeldern und Spaltenüberschriften platziert.

Es wurde darauf geachtet, dass die Daten korrekt eingegeben wurden, um eine genaue Analyse zu ermöglichen.

Nachdem die Daten manuell in die Excel-Tabelle eingetragen wurden, wurden sie auf Richtigkeit und Vollständigkeit überprüft.

Es wurden Validierungsprozesse durchgeführt, um sicherzustellen, dass die Daten gemäß den Geschäftsanforderungen (papiergebundenes Formular) korrekt sind.

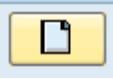

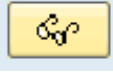


¹⁴ Es ist der Datentyp eines Feldes bzw. dessen mögliche Werte in SAP.

Entwurf der Benutzer Oberfläche (GUI)

Eingabe Feld für Benutzende.

Benutzername. Dieses Feld soll immer eingeblendet sein. Auf der rechten Seite befinden sich fünf Buttons für Optionen, die der Benutzende auswählen kann. Diese Buttons sind Anlegen, Ändern, Anzeigen, Anlegen mit Vorlage, Löschen.

Funktionalität der Buttons:

	Anlegen	Erstellt eine neue Berechtigung für die eingegebene Benutzerkennung.
	Ändern	Aktualisiert die Berechtigung für die eingegebene Benutzerkennung.
	Anzeigen	Zeigt die Berechtigung des ausgewählten Benutzenden an, ohne Bearbeitungsmöglichkeit der Felder.
	Anlegen mit Vorlage	Mit Hilfe dieser Option kann ein Benutzende eine Berechtigung für einen neuen Benutzende basierend auf einer vorhandenen Nutzerberechtigung anlegen.
	Löschen	Entfernt die Berechtigung der ausgewählten Benutzender und blendet alle Felder aus.

Ausgabefeld für die Stammdaten des Benutzenden

 Benutzernummer

 Organisationsnummer

 Nachname

 Vorname

 E-Mail

 Austrittsdatum

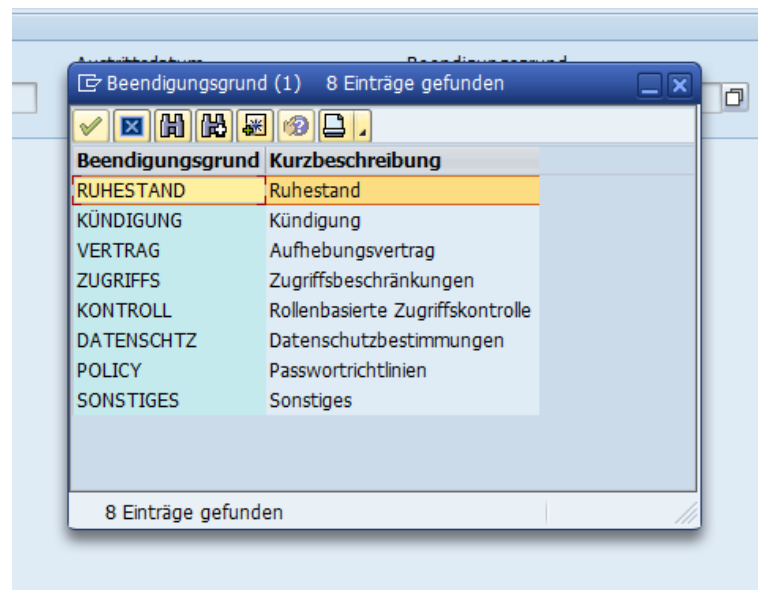
 Endgültig (ja/nein)

 Beendigungsgrund

Konzept zur Sichtbarkeit der Felder und Buttons:

- ⚙️ Beim Start des Programms sind alle Eingabe- und Ausgabefelder ausgeblendet, außer das Eingabefeld für die Benutzende, da dieses erforderlich ist, um eine neue Benutzende darauf einzutragen.
- ⚙️ Nach dem Klick auf „Anlegen“ werden die alle Eingabe- und Ausgabefelder sichtbar. Dafür werden keine Informationen abgerufen (ohne Informationserhaltung).
- ⚙️ Beim Klick auf „Ändern“ werden alle Ausgabefelder sichtbar und die Information und die Berechtigungen des ausgewählten Benutzenden werden aus der Datenbank gelesen.
- ⚙️ Nach dem Klick auf „Anlegen mit Vorlage“ wird eine Modaler Dialog (Pop-up Fenster) angezeigt, damit ein angelegter Benutzende eingetragen werden kann. Durch diese Option werden die alle Berechtigungen der Vorlagebenutzende abgerufen und vorausgefüllt
- ⚙️ Bei Auswahl von „Löschen“ oder „Anzeigen“ werden alle Felder ausgeblendet. Die entsprechenden Informationen und freigeschaltete Berechtigungen in SAP werden wie bei „Ändern“ abgerufen.
- ⚙️ Für Felder mit Namen Benutzende, Beendigungsgrund und Austrittsdatum wurde eine F4 Hilfe¹⁵ hinzugefügt, um eine Liste aller möglichen Eingabe anzeigen zu lassen. Ein Beispiel wurde im folgende dargestellt.

¹⁵ F-4 Hilfe (Eingabehilfe) ist eine Funktion des SAP-Systems, um Eingabe zu erleichtern und Benutzender zu helfen. Dadurch wird eine Liste aller möglichen Eingabewerte für ein Dynpro-Feld angezeigt.



Projektentwicklung und Implementierungsphase

Transportieren Daten in SAP

Um die erforderliche Komponententypen in SAP System zu sammeln, wurden die Komponenten der Strukturen manuell in SAP importiert. Dazu wurde ein Datentyp angelegt.

Dazu wurde für jeden Abschnitt ein Datentyp/eine Struktur angelegt. Anschließend wurde für jede Komponente die entsprechende Struktur hinzugefügt.

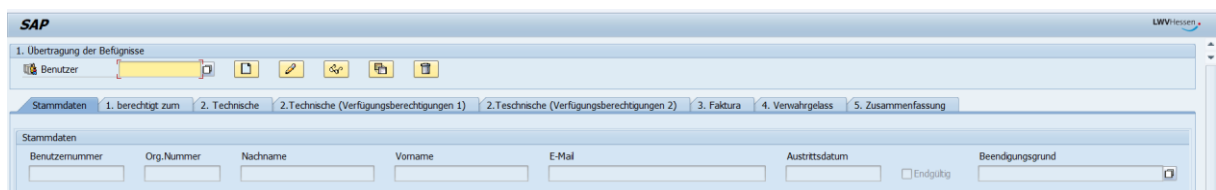
Jedes Rollen wurde in der Tabelle agr_define gefunden und in der Excel Tabelle manuell eingetragen.

Erstellen eine Tabelle zum Verknüpfen von Rollen

Zum Verknüpfen von Rollen wurde eine Tabelle /lwvh/bf_dyn_agr in SAP angelegt mit drei Elementen agr_name (aus agr_define), Tabname (gespeichert in Tabelle DD02L) und Fieldname (gespeichert in Tabelle DD03L). Ausführliche Information befindet sich im [Anhang 9.3](#) auf Seite 32. Abschließend erfolgte eine Fehlerüberprüfung, nachdem alle Rollen in die Tabelle eingetragen wurden.

Erstellen GUI

Danach wurde die Benutzeroberfläche (GUI) erstellt. Bei der Erstellung der GUI wurde auf Benutzerfreundlichkeit geachtet. Es wurde ein Abschnitt zu dem Benutzeranmeldung hinzugefügt, damit der Benutzende Autorisierungsvorgänge durchführen kann. Dieser Abschnitt umfasst die folgenden Funktionen: Benutzende Anlegen, Benutzende Ändern, Benutzende Anzeigen, Benutzende mit Vorlage Anlegen und Benutzende Löschen. Diese Funktionen wurden bereits in der Konzeptphase bereitgestellt.



Die GUI wurde in acht Abschnitte unterteilt: Stammdaten, berechtigt zum, Technische, Technische (Verfügungsberechtigungen 1), Technische (Verfügungsberechtigungen 2), Faktura und Verwahrtgelass. In der Folge wurde auch eine Zusammenfassung hinzugefügt, die das Gesamtbild zeigt. So erhält der Benutzende einen Überblick.

Entwicklung der Methoden

Die Entwicklung der Methoden begann mit einer umfassenden Analyse der Anforderungen und Prozesse, um das Projekt zu schaffen. Anschließend wurde die Entwicklung der Methoden in vier Teile unterteilt und implementiert. Dies sind: Datenbankverarbeiten und SQL-Abfragen, Auslesen der GUI und Anzeigen auf der GUI, ausgewählte Daten anzeigen und Smart Forms. Die entsprechende [Quellcode](#) befindet sich im Angang 9.10 auf Seite 40.

Datenbankverarbeitung und SQL-Abfragen

Im Rahmen des Projekts wurde eine interne Datenbanktabelle für die Speicherung Benutzerbezogener Informationen erstellt, um effiziente SQL-Abfragen zu ermöglichen. Hierbei wurden sämtliche Aspekte berücksichtigt, um die Abfragen sowohl effektiv als auch schnell zu gestalten. Übrigens wurde es das Verfahren Parallel Cursor¹⁶ wie im folgende Code Abschnitt veranschaulicht, implementiert, um die Performance des Programmes zu verbessern und eine bessere Leistung zu erreichen.

METHOD matching_rolle.

DATA:

lv_dynpro_name TYPE char40,

lv_index TYPE i.

* SQL Abfrage und Sammeln die entsprechende Infos in einer Tabelle

¹⁶ Mit parallel Cursors können komplexe Abfragen effizienter bearbeitet werden. Dies ist besonders nützlich in Umgebungen mit hohem Datenvolumen und komplexen Anforderungen an die Datenverarbeitung. Dadurch wird eine bessere Leistungsfähigkeit der SQL-Abfrage erreicht.

```

SELECT *
  FROM /lwwh/bf_dyn_agr
 INTO TABLE lt_dyn_agr.

* Parallel Cursor zu benutzen, sortieren die Tabelle
  SORT lt_get_rollen BY agr_name ASCENDING.
  SORT lt_dyn_agr BY agr_name ASCENDING.

  LOOP AT lt_get_rollen INTO ls_get_rollen.

* Read Tabelle die Rollen zu finden
    READ TABLE lt_dyn_agr INTO ls_dyn_agr WITH KEY agr_name = ls_get_rollen-
agr_name BINARY SEARCH .

* Wenn es erfolgreich, bestimmen index der Rollen
    IF sy-subrc EQ 0.
      lv_index = sy-tabix .
    ENDIF.

* Mit bestimmten Rollen suchen die Rollen in einer anderen Tabelle
    LOOP AT lt_dyn_agr INTO ls_dyn_agr FROM lv_index.

      IF ls_dyn_agr-agr_name NE ls_get_rollen-agr_name.
        EXIT.
      ELSE.
        CONCATENATE ls_dyn_agr-tabname ls_dyn_agr-
fieldname INTO lv_dynpro_name SEPARATED BY '-'.

        ls_dynpfields-fieldname = lv_dynpro_name .
        ls_dynpfields-fieldvalue = 'X' .

        APPEND ls_dynpfields TO lt_dynpfields.
        CLEAR ls_dynpfields .
      ENDIF.
    ENDLOOP.
  ENDLOOP.
ENDMETHOD.

```

Ablesen GUI und Anzeigen auf der GUI

Des Weiteren werden alle vom Benutzende eingegebenen Rollen automatisch in der vordefinierten Struktur von SAP gespeichert. Die im SAP-System hinterlegten Rollen eins

Benutzenden sowie ihre Bezeichnungen werden über SAP aufgerufen. Danach werden die entsprechenden Informationen in einer internen Tabelle durch eine Funktionsbaustein im GUI dargestellt.

Ausgewählte Daten anzeigen

Alle relevanten Informationen werden in einer internen Tabelle gesammelt und im Bereich "Zusammenfassung" präsentiert.

Smart Forms

Durch die Implementierung eines PDF-Generators mit Hilfe Smart Forms wurden sämtliche ausgewählten Rollen und Stammdaten an das papiergebundene Formular angepasst und dem Benutzende zur Verfügung gestellt. Zudem ist nun das Drucken der PDF-Dokumente möglich. Ein Teil des Formulars ist im Folgenden beigefügt.

Mitteilung über die Übertragung von Befugnissen

* im Anordnungswesen gemäß Nr. 9.2 GA Anordnungswesen und

* zur Einrichtung der Zugriffsberechtigungen im SAP-System im LWV Hessen¹

1. Übertragung der Befugnisse

Stammdaten

Die nachfolgend aufgeführte Person

Benutzernummer:	Organisationsnummer:	Austrittsdatum:	Endgültig:	Beendigungsgrund:
	102.007			
Nachname:	Vorname:	E-Mail:		
102.0	Azubi	@lww-hessen.de		

ist für ihren/seinen Aufgabenbereich berechtigt zum

<input type="checkbox"/>	unbegrenztem Erfassen ²
<input type="checkbox"/>	unbegrenztem Anordnen ²
<input type="checkbox"/>	von Kassenanordnungen
<input type="checkbox"/>	im analogen/papier-gebundenen Freigabesystem
<input type="checkbox"/>	im SAP-System mit den entsprechenden technischen Zugangsberechtigungen
<input type="checkbox"/>	Zugriff auf das SAP-System

1.1 Bestätigung und Unterschriftsprobe der/des Befugten

(Nur erforderlich, wenn Erfassungs- und/oder Anordnungsbefugnis beantragt wird)

Datum:

Unterschrift: _____

1.2 Bestätigung der/des Vorgesetzten

(Anmerkung: Für Erfassungs- oder Anordnungsbefugnis ist die Unterschrift des Vorgesetzten gemäß Nr. 9.2 GA Anordnungswesen erforderlich)

Datum:

Unterschrift: _____

¹SAP-Zugriff gilt nur in Verbindung mit in ANLEI vorhandenen, aktiven Mitarbeiterstammdaten und Organisationsnummer. Wenn nicht vorhanden, oder nicht aktuell, ist ein ANLEI-Antrag zusätzlich auszufüllen: Nur Stammdaten oder ein kompletter Antrag zur Einrichtung der ANLEI-Benutzerrechte.

²Es kann beides angekreuzt werden. Systemseitig ist dabei sichergestellt, dass beide Berechtigungen nicht auf ein und dieselbe Anordnung angewandt werden.

2.3 c) Rechnungswesen

☐ Erfassen / Feststellen
☐ Zentrale ANBU
☐ Stammdaten
☐ LOGA Schnittstelle
☐ Beauskunftung HÜL

☐ Anordnen
☐ Dezentrale ANBU
☐ Steuern
☐ Sofia Schnittstelle
☐ Übertragung Obligo

☐ Controlling
☐ Kassenbuch
☐ Darlehenssachbearbeitung
☐ Darlehen Anordnen
☐

2.3 d) Sonstiges

☐ Support

☐ Revision

☐ Hotline

2.3 e) Nur Beauskunftung

☐ FI-Hauptbuch
☐ Finanzrechnung / Haushalt
☐ Beauskunftung alle Module

☐ PSCD-Vertragskontokorrent
☐ Darlehen
☐ Kassenbuch

☐ Controlling Info
☐ Anlagenbuchhaltung

Qualitätsmanagement

Testphase

Die Fehleranalyse im Qualitätsmanagement ist ein wichtiger Schritt, um Schwachstellen in der Software zu identifizieren und korrigieren. In diesem Prozess werden verschiedene Arten von Tests durchgeführt, welche Unit-Tests, Integrationstests und Systemtests sind.

Bei der Fehleranalyse können verschiedene Arten von Fehlern auftreten, wie beispielsweise Probleme mit globalen Deklarationen oder Variablenbenennungen. Es ist wichtig, diese Fehler zu dokumentieren und entsprechende Maßnahmen zur Behebung einzuleiten.

Unit-Tests

Die Unit-Tests wurden sofort nach der Implementierung durchgeführt, um Fehler frühzeitig zu erkennen. Dabei wurden fachliche und technische Fehler geprüft. Auf die Verständlichkeit des Codes in Bezug auf die Benennung von Variablen und der Methoden wurde gemäß Coderichtlinie des LWV Hessen geachtet.

Integrationstests

Integrationstests wurden Schritt für Schritt getestet, um sicherzustellen, dass die einzelnen Codezeilen gemeinsam funktionieren.

Dabei wurde die Funktionalität von verschiedenen Komponenten des Systems geprüft, um sicherzustellen, dass Daten korrekt übergeben werden und keine Schnittstellenprobleme auftreten.

Systemtests

Es wurden Szenarien erstellt, die typische Benutzerinteraktionen und Anwendungsfälle abbilden. Dabei wurden die erstellten Szenarien nicht nur per White-Box Methode, sondern auch mit der Black-Box Methode getestet, um sicherzustellen, dass das System die erwarteten Ergebnisse liefert und die Anforderungen erfüllt. Dadurch ist die Zuverlässigkeit und Stabilität des Systems gewährleistet.

White-Box-Tests:

Mit Hilfe von Break-Points und Szenarien wurden die Quellcodes analysiert.

Dank der Durchführung der Tests wurde sichergestellt, dass alle internen Komponenten und Logikpfade ordnungsgemäß funktionieren.

Black-Box-Tests:

Nach Fertigstellung der Anwendung fanden Black-Box-Tests mit Szenarien durch Kollegen statt, um sicherzustellen, dass das System korrekt funktioniert und Benutzerfreundlich ist.

Fehleranalyse

Die Testergebnisse wurden analysiert, um Schwachstellen zu identifizieren und den Testprozess zu optimieren sowie die Softwarequalität kontinuierlich zu verbessern.

Die Kombination von White-Box- und Black-Box-Tests wurde gewählt, um eine umfassende Testabdeckung zu gewährleisten und sowohl interne als auch externe Aspekte des Systems zu überprüfen.

Folgende Fehler sind bei der Fehleranalyse aufgekommen:

- ⚙ Globale Deklarationen waren zu viel und wurden reduziert.
- ⚙ Variablenbenennungen sind laut „Dokumentation und Richtlinien“ für ABAP Entwicklerrichtlinien angepasst.
- ⚙ Die Rollen wurden beim Wechsel zwischen den Funktionen markiert. Die Tabs der GUI wurden für jede Rollen aktualisiert. Dazu wurde eine Globale Deklaration mit dem Namen „gv_ok_code“ deklariert und in der PAI, je nach Tab, importiert und verwendet.
- ⚙ Der Funktionsbaustein zum Lesen von aufgezeichneten Rollen funktionierte nur einmal pro Programmaufruf. Dafür wurde die Method „set_tab“ erstellt. Anschließend wurde je nach dem Tab ein Funktionsbaustein, nämlich „DYNP_UPDATE_FIELDS“, in Method „update_dyn“ aufgerufen.

Abschlussfazit

Mit der Abschlussfazit und Projektabnahme wurden Projektziele abgeglichen, um erledigte, noch offene und unmögliche Punkte festzustellen.

Fachliches und Persönliches Fazit

Die Projektleitung und die Ergebnisse wurden aus verschiedenen Perspektiven überprüft. Dazu gehört die Beantwortung folgender Fragen:

- ⚙️ Wurden die Projektziele erfolgreich erreicht?
- ⚙️ Wie zufrieden ist der Auftraggeber mit dem Ergebnis?
- ⚙️ Entspricht das Ergebnis den Erwartungen bezüglich der Qualität und der Zeit?
- ⚙️ Wie zufrieden sind Mitarbeitende?

Aus meiner Aufsicht kann ich betonen, dass die Projektziele erfolgreich erreicht wurden. Der Auftraggeber äußerte sich in Zufriedenheit über das Ergebnis, das Zeitmanagement und der Qualität des Projekts. Die Zufriedenheit der Mitarbeiter bedarf weiterer Evaluierung und wird beobachtet, um Verbesserungsmöglichkeiten zu identifizieren.

Lessons-Learned

Lessons-Learned definiert hier durch die Methode des Sammelns der Erkenntnisse während des Projektes. Das zielt darauf ab, dass man in neuen Projekten Fehler vermeiden, Projekt-Qualität steigern und Chancen ergreifen kann.

In einer Besprechung zum Thema: „Was haben wir gelernt?“ wurden folgende Punkte besprochen:

- ⚙️ Gleichzeitig nur an einer Aufgabe arbeiten. Einerseits musste ich mich mit Projektarbeit befassen, andererseits gab es die Prüfungsvorbereitung. Es wäre besser, wenn ich ohne Unterbrechungen arbeiten könnte.
- ⚙️ Arbeitsplatz einrichten. Ich habe mit der Projektarbeit KW 9 begonnen, musste aber 3 Tage später aufgrund eines Bausteines der Prüfungsvorbereitung beim Bildungsträger das Praktikum pausieren. Nach einer einmonatigen Pause fällt es mir auch schwer, wieder einzusteigen. Um effektiv arbeiten zu können, ist es wichtig, nicht nur einen geeigneten Arbeitsplatz, sondern eine geeignete Arbeitszeit einzurichten

Projektübergabe

Die Projektabnahme und Produktübergabe wurden organisiert. Das Projektziel wurde erreicht und das Projekt wurde den Anforderungen entsprechend zur vollen Zufriedenheit des Auftraggebers umgesetzt.

Dabei wurden alle Aufgaben erledigt oder offene Punkte identifiziert und in einer Liste festgehalten, die weiterabgearbeitet wird, um das Projekt vollständig zu gewährleisten.

Alle Aufgaben wurden erledigt oder offene Punkte identifiziert und in einer Liste festgehalten, die weiterbearbeitet wird, um das Projekt vollständig abzuschließen.

🔧 erledigt:

- Database Tabellen mit Attributen Namen und Strukturen
- GUI Funktionalitäten: Anzeigen, Anlegen, Anlegen mit Vorlage und Ändern
- Datenbankmodell erstellt
- Rollenverknüpfung durchgeführt
- SQL-Abfragen mit oder ohne parallelen Cursor

🔧 offen:

- Speichern-Funktion implementieren
- Druckfunktion hinzufügen
- Berechtigungsadministration abschließen

🔧 nicht möglich: Es gibt keine unmöglichen Aufgaben bei der Übergabe des Projekts.

Nach erfolgreichem Abschluss der Testphase wurde die neue Transaktion in SAP integriert und die Mitarbeitende des LWV Hessen zur Nutzung bereitgestellt. Das Benutzerhandbuch wurde erstellt, um eine reibungslose Einführung sicherzustellen.

Literaturverzeichnis

ABAP Die offizielle Referenz, Keller Horst, Rheinwerk Publishing, 4.Auflage, 2016.

SAP, BC410 Benutzerdialoge mithilfe von Klassischen Dynpros programmieren, Teilnehmerhandbuch, 2011.

Dokumentation und Richtlinien für ABAP Entwicklerrichtlinien, Handbuch, LWV Hessen, Version 4.0, 2019.

https://help.sap.com/doc/saphelp_gbt10/1.0/de-DE/4a/44b861954c0453e10000000a421937/content.htm?no_cache=true. (17. 04 2024). Von https://help.sap.com/doc/saphelp_gbt10/1.0/de-DE/4a/44b861954c0453e10000000a421937/content.htm?no_cache=true abgerufen

https://help.sap.com/saphelp_autoid2007/helpdata/de/fc/eb2e97358411d1829f0000e829fbfe/content.htm?no_cache=true. (09. 04 2024). Von https://help.sap.com/saphelp_autoid2007/helpdata/de/fc/eb2e97358411d1829f0000e829fbfe/content.htm?no_cache=true abgerufen

<https://rz10.de/knowhow/sap-netweaver-application-server-abap-as-abap/>. (09. 04 2024). Von <https://rz10.de/knowhow/sap-netweaver-application-server-abap-as-abap/> abgerufen

<https://www.gambit.de/wiki/unterschiede-sap-r3-ecc-und-s4hana/>. (9. 4 2024). Von <https://www.gambit.de/wiki/unterschiede-sap-r3-ecc-und-s4hana/> abgerufen

<https://www.sap.com/germany/about/what-is-sap.html>. (9. 04 2024). Von <https://www.sap.com/germany/about/what-is-sap.html> abgerufen

Model-view-controller - *Wikipedia*. (02. 04 2024). Von https://de.wikipedia.org/wiki/Model_View_Controller abgerufen

MVC Design Pattern | *SAP Help Portal*. (02. 04 2024). Von https://help.sap.com/docs/SAP_NETWEAVER_700/12aa7f056c531014aa5bca7aee037e55/4c3fd332a2a54f8be10000000a42189b.html abgerufen

Glossar

ABAP	Advanced Business Application Programming. ABAP ist eine Programmiersprache der Softwarefirma SAP und wird für SAP entwickelt.
Black-Box-Tests	Werden durchgeführt, ohne Kenntnisse über die interne Struktur des Systems zu haben. Stattdessen wird das System als Black-Box betrachtet und die Tests konzentrieren sich auf die Eingaben, Ausgaben und das Verhalten des Systems.
ERM	Entity-Relationship-Modell
GUI	Graphical User Interface
HANA DB	High-performance ANalytic Appliance Datenbank, welche von SAP in Zusammenarbeit mit dem Hasso-Plattner-Institut und der Stanford Universität für das SAP-System entwickelt wurde.
Integrationstests	prüfen, ob die Einheiten des Systems korrekt miteinander interagieren und wie sie als Gruppe funktionieren.
LWV	Landeswohlfahrtsverband Hessen
MVC	Das Model-View-Controller.
P01	Abkürzung für Implementation der Methoden. Die Methoden befinden sich in diesem Abschnitt.
Parallel Cursor	Mit parallel Cursors können komplexe Abfragen effizienter bearbeitet werden. Dies ist besonders nützlich in Umgebungen mit hohem Datenvolumen und komplexen Anforderungen an die Datenverarbeitung. Dadurch wird eine bessere Leistungsfähigkeit der SQL-Abfrage erreicht.
PAI	Process After Input wird ausgelöst bei Aktion vom User auf dem Bildschirm

PBO	Process Before Output, wird von der ABAP-Laufzeitumgebung vor dem erscheinen Bildschirm an die Präsentationsschicht ausgelöst.
Smart Forms	SAP Smart Forms ist ein Tool zum Erstellen der Formulare in SAP-Systemen.
Systemtests	prüfen das gesamte System aus der Perspektive des Endbenutzendes, um sicherzustellen, dass es die Anforderungen erfüllt und korrekt funktioniert.
Unit-Tests	sind Tests auf Codeebene, die einzelne Einheiten (Funktionen, Methoden oder Klassen) isoliert prüfen, um sicherzustellen, dass sie korrekt funktionieren.
UML	Unified Modeling Language.
Use-Case	Als Anwendungsfalldiagramm benennt auch und dient Interaktionsmöglichkeiten zwischen einem Benutzende und einem System.
VHK	Verbandshauptkasse.
White-Box-Tests	werden durchgeführt, um die interne Struktur und das Verhalten des Systems zu überprüfen. Dabei werden Kenntnisse über den internen Aufbau des Codes verwendet, um effektive Tests zu erstellen.

Anhänge

Lastenheft

Die Anwendung muss folgende Anforderungen erfüllen:

Verarbeitung der Daten

- ⚙ Die Anwendung muss in der Lage sein, Genehmigungsanträge aus einer Datenbank auszulesen und zu verarbeiten.
- ⚙ Es sollen Korrektheit der Einträge während der Verarbeitung der Anträge durchgeführt werden.
- ⚙ Eine Übertragung der Informationen der Einträge in einer Datenbank ist notwendig.

Darstellung der Daten

- ⚙ Die Anwendung muss eine Benutzerfreundliche Oberfläche bereitstellen, die das Ausfüllen und Einreichen von Genehmigungsanträgen ermöglicht.
- ⚙ Eine Druckmöglichkeit in PDF sollen bereitgestellt werden.

Sonstige Anforderungen

- ⚙ Die Anwendung muss in SAP als Transaktion erreichbar sein.
- ⚙ Ein Benutzerfreundliches GUI-Interface wird entwickelt, das eine einfache Bedienung und Navigation anbieten soll.

Zusammenfassung

- ⚙ Das Lastenheft stellt die grundlegenden Anforderungen und Spezifikationen für die Entwicklung eines automatisierten Genehmigungsprozesses in SAP dar.

Pflichtenheft

Bereits wurden Projektumsetzung, Projektziel, Projektanalyse und Ressourcenplanung in den entsprechenden Abschnitten genannt. Aus diesem Grund wurde hier nicht mehr erwähnt.

Plattform

- ⚙ Zur Entwicklung der Anwendung wird die SAP-Entwicklungsumgebung verwendet.
- ⚙ Die Anwendung wird in ABAP (Advanced Business Application Programming) programmiert.
- ⚙ Es wird SAP NetWeaver for ABAP 7.52 als Application Server verwendet.
- ⚙ Die Anwendung muss innerhalb des LWV-Netzwerks in SAP erreichbar sein.
- ⚙ Als Betriebssystem für den Rechner wird Windows-10 eingesetzt.

Datenbank

- ⚙ Die Datenbank wird in der HANA DB in SAP-System gehostet.
- ⚙ Alle Daten der Anwendung werden in der internen SAP-HANA-Datenbank gespeichert.
- ⚙ Datenbankmodell ist als [Entity-Relationship-Modell \(ERM\)](#) im Anhang 9.9: auf Seite 39 dargestellt.

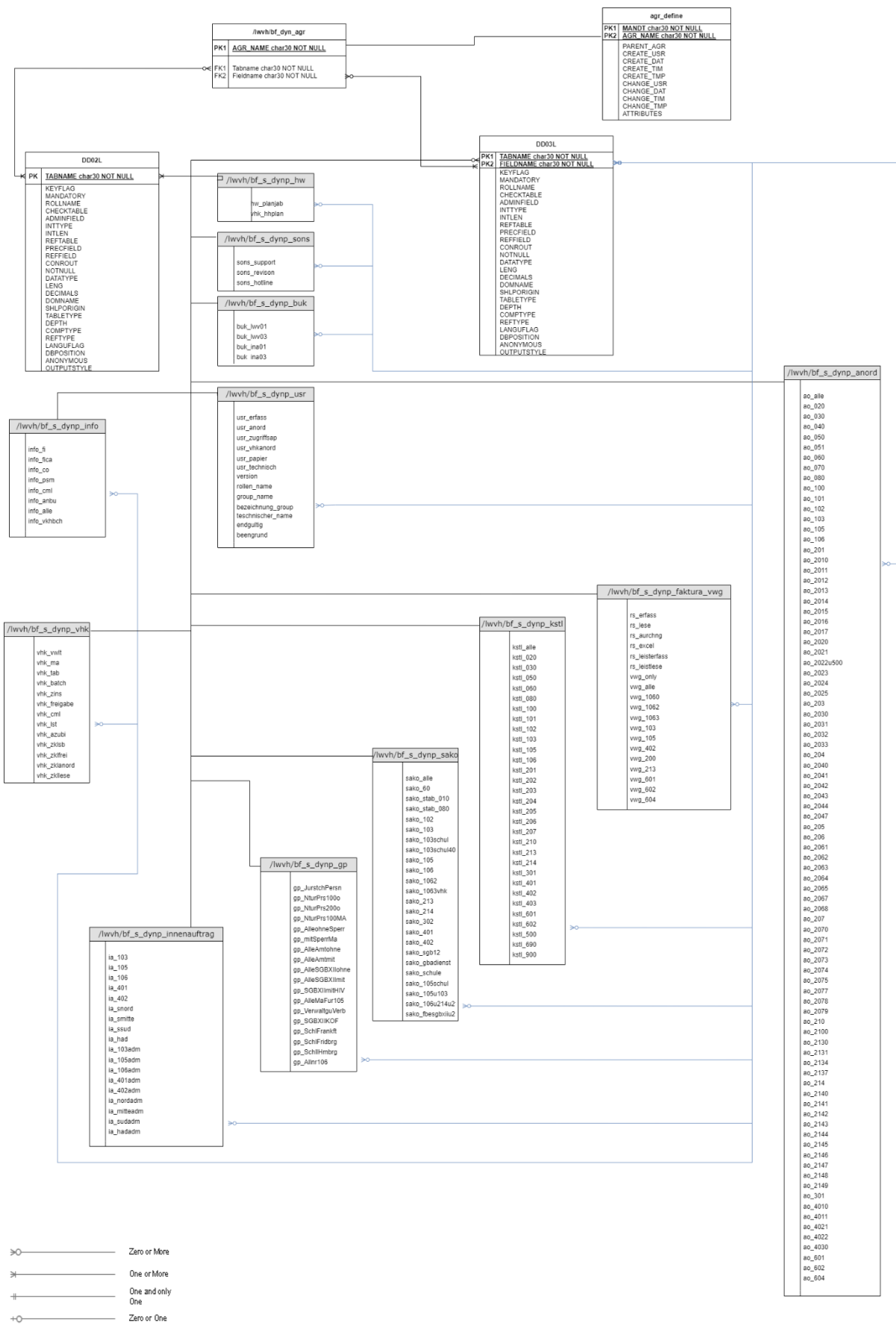
Oberfläche

- ⚙ Die Oberfläche wird in SAP GUI (Graphical User Interface) erstellt.
- ⚙ Die Benutzeroberfläche wird mit Dynpro-programmierung entwickelt.

Geschäftslogik

- ⚙ Zur Entwicklung der Geschäftslogik verwendet ABAP Objects.
- ⚙ Die Tests werden in SAP durchgeführt.
- ⚙ Die Erstellung und Verarbeitung von Excel-Dateien erfolgten direkt in SAP.

Datenbankmodell



Ein Teil des Antrags

Mitteilung über die Übertragung von Befugnissen

- im Anordnungswesen gemäß Nr. 9.2 GA Anordnungswesen und
- zur Einrichtung der Zugriffsberechtigungen im SAP-System im LWV Hessen¹

1. Übertragung der Befugnisse

Stammdaten

Die nachfolgend aufgeführte Person

Benutzernummer:	Organisationsnummer:	Austrittsdatum:	<input type="checkbox"/> endgültig
		Beendigungsgrund:	
Nachname:		Vorname:	
E-Mail:			

ist für ihren/seinen Aufgabenbereich berechtigt zum

- ☐ unbegrenztem Erfassen²
- ☐ unbegrenztem Anordnen²
- von Kassenanordnungen
- ☐ im analogen/papier-gebundenen Freigabesystem
- ☐ im SAP-System mit den entsprechenden technischen Zugangsberechtigungen.
- ☐ Zugriff auf das SAP-System.

¹SAP-Zugriff gilt nur in Verbindung mit in ANLEI vorhandenen, aktiven Mitarbeiterstammdaten und Organisationsnummer. Wenn nicht vorhanden, oder nicht aktuell, ist ein ANLEI-Antrag zusätzlich auszufüllen: Nur Stammdaten oder ein kompletter Antrag zur Einrichtung der ANLEI-Benutzerrechte

²Es kann beides angekreuzt werden. Systemseitig ist dabei sichergestellt, dass beide Berechtigungen nicht auf ein und dieselbe Anordnung angewandt werden.

2.4 Verfügungsberechtigungen

2.4 a) Verfügungsberechtigung für Sachkonten

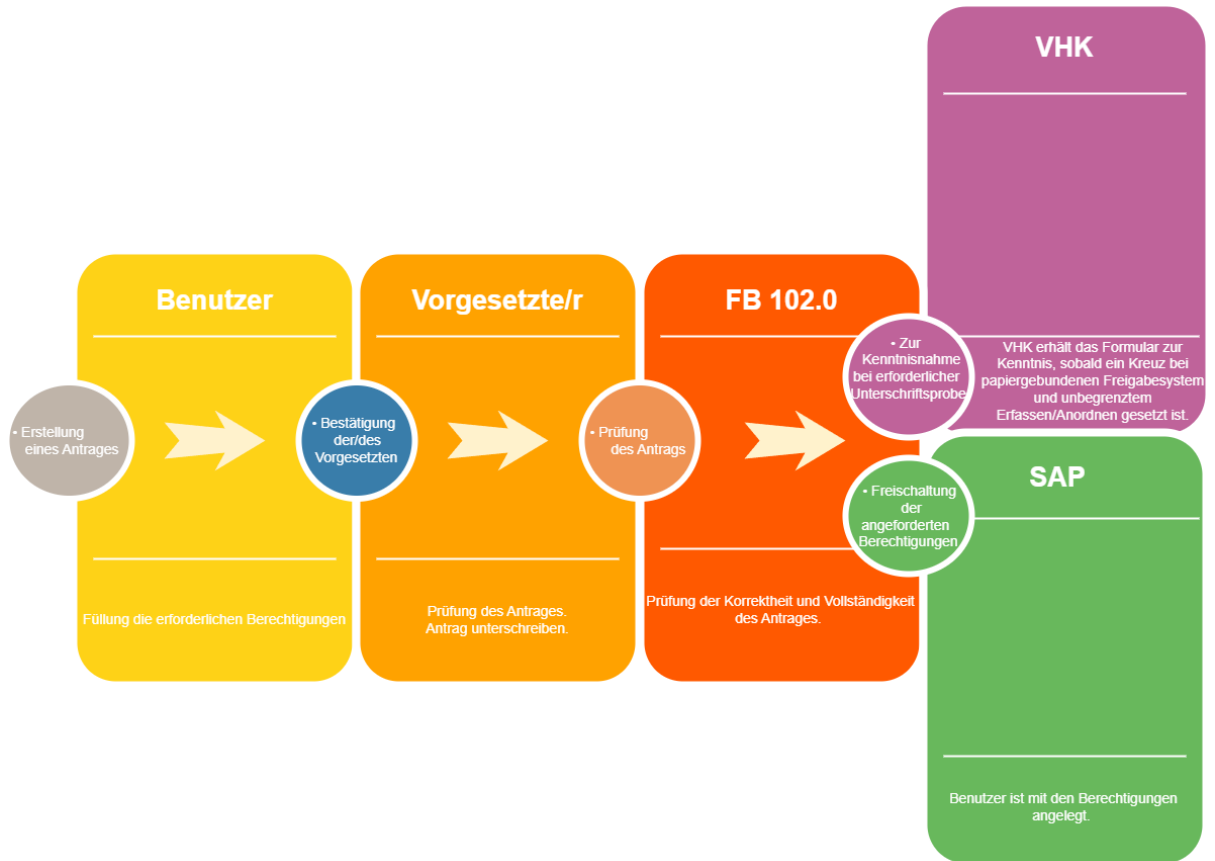
Erläuterung zur Aufteilung der jeweils hinterlegten Konten:

Sachkonten sind bei den einzelnen Verfügungsberechtigungen nicht mehrfach hinterlegt, d. h. ein Anwender, der Zugriff auf alle Konten von FB 103 erhalten soll, benötigt alle Punkte, in denen der FB 103 genannt ist.

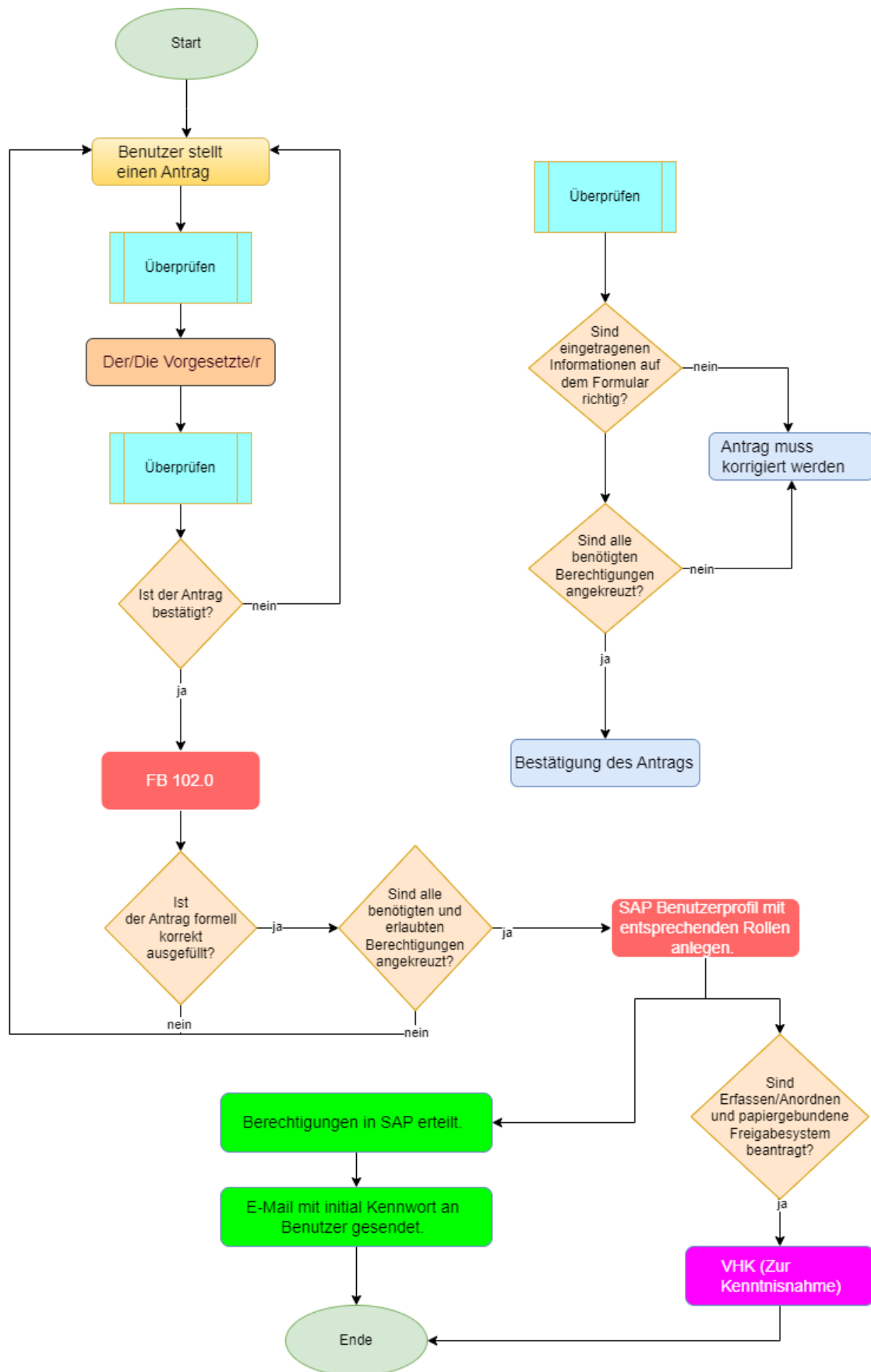
<input type="checkbox"/> Alle (nur FB106; 020)	<input type="checkbox"/> Stabsstelle 010 und LD/EB	<input type="checkbox"/> nur FB 102
<input type="checkbox"/> nur FB 060	<input type="checkbox"/> FB 103 und Schulen	<input type="checkbox"/> FB 103 und Schulen und FB 402
<input type="checkbox"/> nur FB 103	<input type="checkbox"/> Schulen und FB 105	<input type="checkbox"/> nur FB 105 und FB 103
<input type="checkbox"/> nur Schulen	<input type="checkbox"/> nur FB 106	<input type="checkbox"/> nur FB 106.2
<input type="checkbox"/> nur FB 105	<input type="checkbox"/> FB 106, FB 214 und FB 213	<input type="checkbox"/> nur Stabsstelle 080
<input type="checkbox"/> nur FB 106.3 VHK	<input type="checkbox"/> nur FB 214	<input type="checkbox"/> FBe SGB XII, 213 und 214
<input type="checkbox"/> nur FB 213	<input type="checkbox"/> nur BgA Dienstleistungen	<input type="checkbox"/> nur FBe SGB XII
<input type="checkbox"/> nur FB 302	<input type="checkbox"/> nur FB 402	
<input type="checkbox"/> nur FB 401		

Ablaufplan für papierbasierten Prozess

IST-Stand

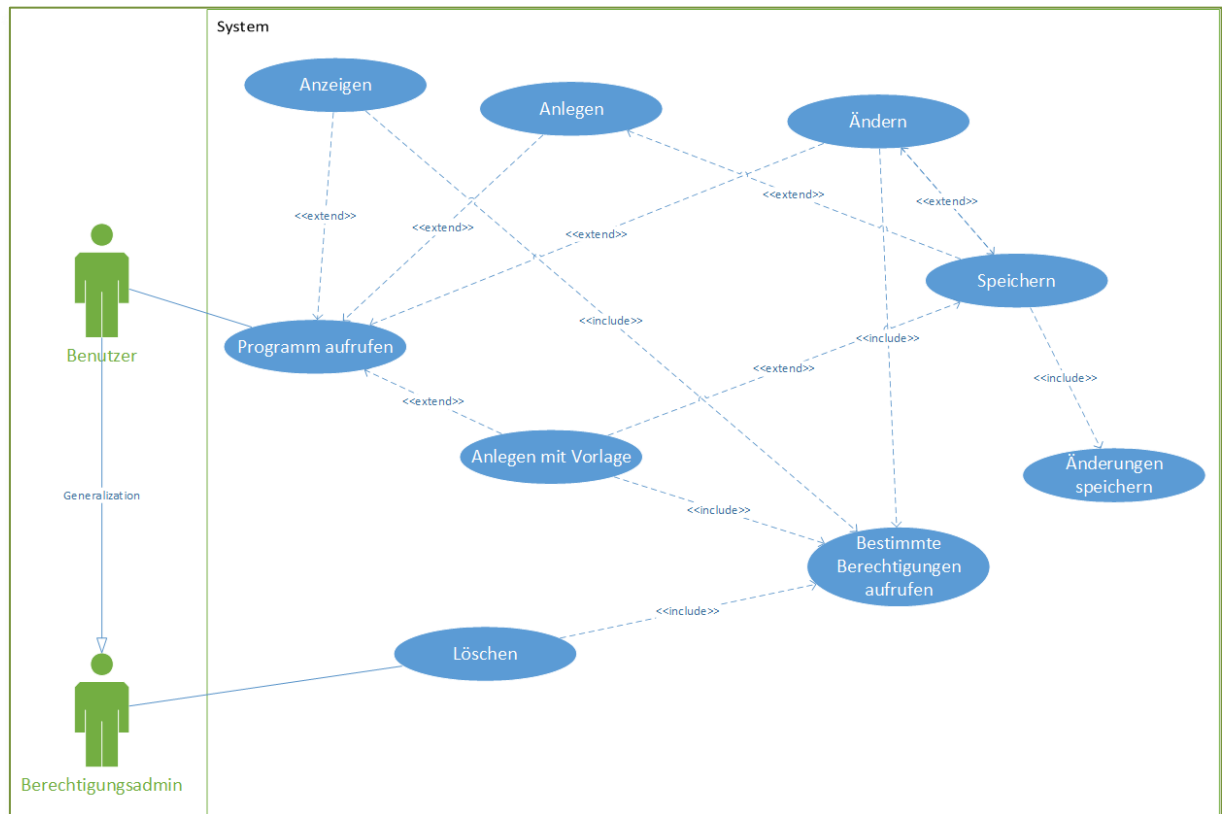


Ausführlicher Ablaufplan

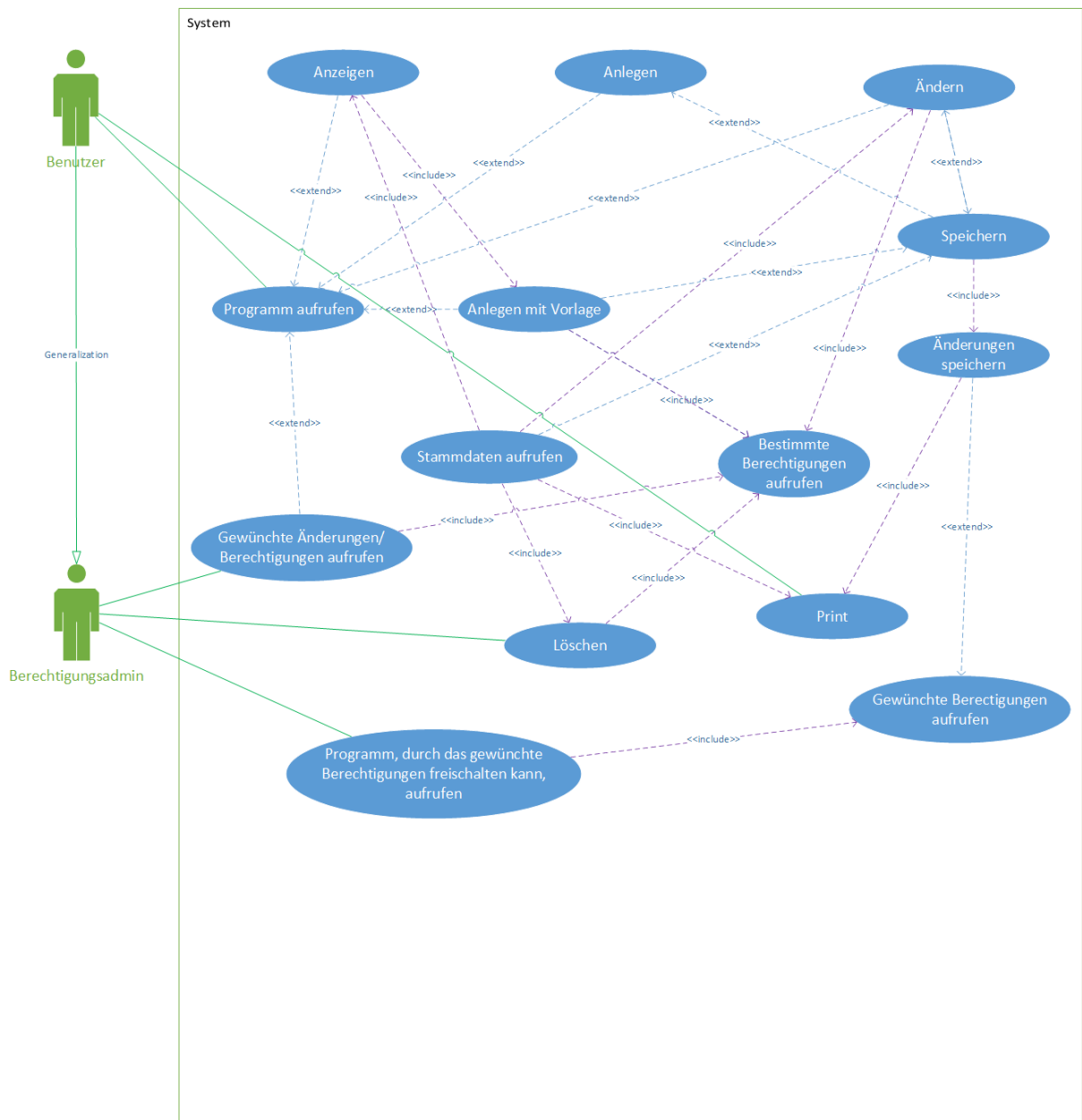


Use-Case Diagramm

Use-Case Diagramm fürs Projekt



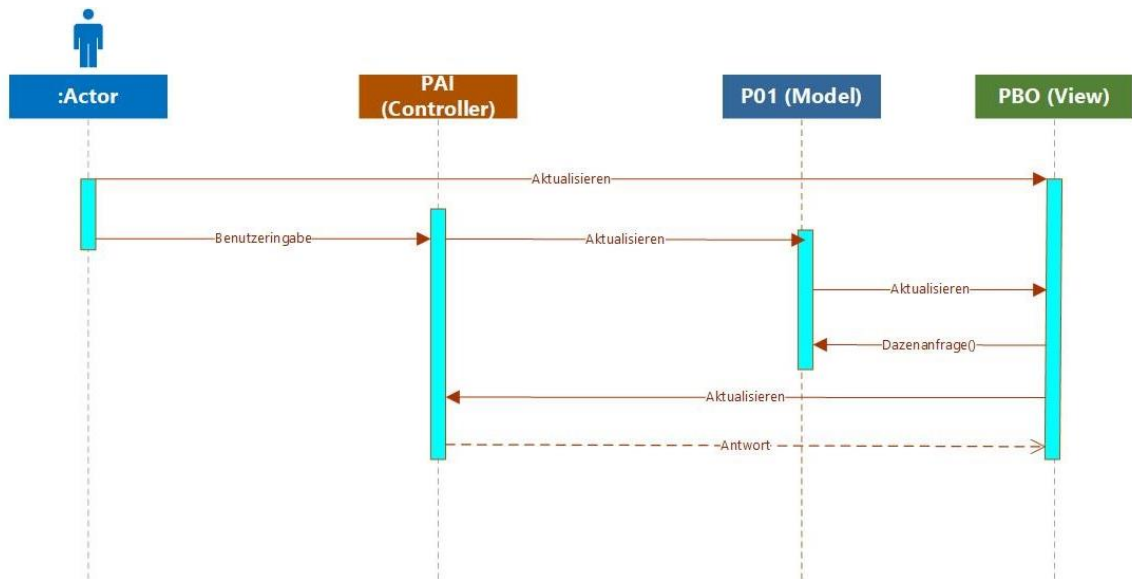
Erweiterte Use-Case-Diagramm



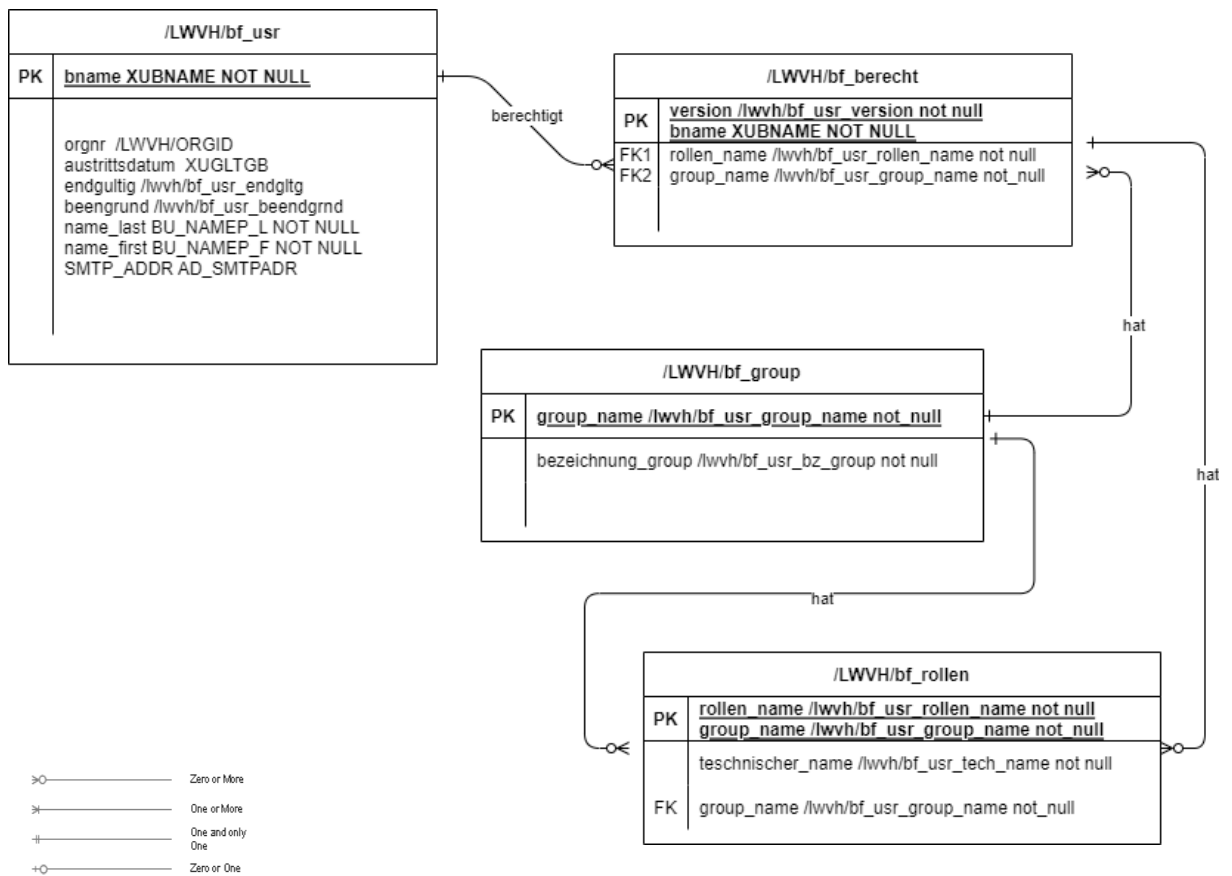
Projektphasen mit Zeitplanung in Stunden

Detaillierte Zeitplanung in Stunden	KW 9	KW 14	KW 15	KW 16	Geplant	Tatsächlich
Projektbeschreibung					5	5
Projektziel- und umfeld	1					
Projektanalyse	0,5					
Ist- und Soll Analyse	1					
Projktbegründung	0,5					
Lastenheft	2					
Konzeption Phase					25	24
Ablaufplan für papierbasierten Prozess	2					
Architekturdesign		1				
Pflichtenheft		2				
Anwendungsfälle (Use Case)	2					
Datenmodell	4					
Erstellung einer Excel-Tabelle für SAP-Daten	5	2				
Entwurf der Benutzeroberfläche (GUI)		6				
Projektentwicklung und Implementierungsphase					32	32
Transportieren Daten in SAP		5				
Erstellen eine Tabelle zum Verknüpfen von Rollen		3				
Erstellen GUI		3				
Entwicklung der Methoden						
Datenbankverarbeiten und SQL-Abfragen		3				
Ablesen GUI und Anzeigen auf der GUI		2	4			
Ausgewählte Daten Anzeigen			4			
Smart Form			4	4		
Qualitätsmanagement					6	7
Unit-Tests		1	1			
Integrationstests				2		
Systemtests				2		
Fehleranalyse				1		
Abschlussfazit					3	3
Fachliches und Persönliches Fazit				1		
Lessons-Learned				1		
Projektübergabe				1		
Erstellen der Dokumentationen	1	1	3	4	9	9
Gesamtdauer	19	29	16	16	80	80

MVC-Sequenzdiagramm



Entity-Relationship-Modell (ERM)



Quellcode

TOP - PROGRAM /LWVH/BERECHTIGUNGFORMULA.

TABLES : agr_users,
/lwvh/bf_dyn_agr, *"Tabelle zur Verknüpfung von GUI-Namen und Rollennamen"*
agr_define, *"Rollen"*
usr01,
/lwvh/orgnrusr,
usr21,
adrp,
/lwvh/bf_s_dynp_usr,
/lwvh/bf_s_dynp_buk,
/lwvh/bf_s_dynp_vhk,
/lwvh/bf_s_dynp_hw,
/lwvh/bf_s_dynp_rchn,
/lwvh/bf_s_dynp_sons,
/lwvh/bf_s_dynp_info,
/lwvh/bf_s_dynp_sako,
/lwvh/bf_s_dynp_kstl,
/lwvh/bf_s_dynp_gp,
/lwvh/bf_s_dynp_anord,
/lwvh/bf_s_dynp_innenauftrag,
/lwvh/bf_s_dynp_faktura_vwg.

DATA: ok_code **TYPE** sy-ucomm,
enter **TYPE** sy-ucomm,
tx_io **TYPE** xubname,
tb_anlegen **TYPE** char1,
tb_andern **TYPE** char1,
"tb_anzeigen" TYPE char1,
tb_anlage_mit_vorlage **TYPE** char1,
tb_loschen **TYPE** char1,
tx_Benutzendennummer **TYPE** usr21-persnumber,
tx_vorlage_io **TYPE** agr_users-uname.

** Save Button Einstellung*

DATA: gv_show **TYPE** xfeld,
gv_edit **TYPE** xfeld,
gv_create **TYPE** xfeld,
gv_save_first **TYPE** abap_bool **VALUE IS INITIAL**,
gv_save_button **TYPE** abap_bool,
gv_ok_code **TYPE** char20 . *"Einstellung der subdyns"*

**data usr*

DATA: tx_bname **TYPE** xubname,
tx_orgnr **TYPE** /lwvh/orgid,
tx_austrittsdatum **TYPE** xugltgb,
cb_endgultig **TYPE** /lwvh/bf_usr_endgltg,
tx_beengrund **TYPE** /lwvh/bf_usr_beendgrnd,
tx_name_last **TYPE** bu_namep_l,
tx_name_first **TYPE** bu_namep_f,
tx_smtp_addr **TYPE** ad_smtpadr.

**constants für subdyn*

```

DATA: gc_dyn_100 TYPE char4 VALUE '0100',
      gc_dyn_110 TYPE char4 VALUE '0110',
      gc_dyn_220 TYPE char4 VALUE '0220',
      gc_dyn_240 TYPE char4 VALUE '0240',
      gc_dyn_241 TYPE char4 VALUE '0241',
      gc_dyn_300 TYPE char4 VALUE '0300',
      gc_dyn_400 TYPE char4 VALUE '0400',
      gc_dyn     TYPE char4.

```

** Die Tabelle für die Rollen auf subscreens*

```

DATA: lt_dynpfields TYPE TABLE OF dynpread,
      ls_dynpfields TYPE dynpread.

```

```

DATA: lt_dynpfields_save TYPE TABLE OF dynpread,
      ls_dynpfields_save TYPE dynpread.

```

** Die Tabelle für bereits eingegebene Rollen*

```

DATA: lt_get_rolle TYPE TABLE OF agr_users,
      ls_get_rolle TYPE agr_users.

```

** F4 Hilfe*

```

* DATA: lt_bname TYPE TABLE OF usr01,
*        ls_bname TYPE usr01.

```

** Die Folgende wurden von SAP automatisch erstellt.*

** &SPWIZARD: FUNCTION CODES FOR TABSTRIP 'TAB_01'*

```

CONSTANTS: BEGIN OF c_tab_01,
            tab1 LIKE sy-ucomm VALUE 'TAB_01_FC1',
            tab2 LIKE sy-ucomm VALUE 'TAB_01_FC2',
            END OF c_tab_01.

```

** &SPWIZARD: DATA FOR TABSTRIP 'TAB_01'*

```

CONTROLS: tab_01 TYPE TABSTRIP.
DATA: BEGIN OF g_tab_01,
      subscreen LIKE sy-dynnr,
      prog      LIKE sy-repid VALUE '/LWVH/BERECHTIGUNGFORMULA',
      pressed_tab LIKE sy-ucomm VALUE c_tab_01-tab1,
      END OF g_tab_01.

```

```

CONTROLS container TYPE TABSTRIP.

```

```

CLASS lcl_berechtigungformula DEFINITION.
PUBLIC SECTION.

```

```

METHODS: get_stammdaten
IMPORTING i_tx_io TYPE xubname,
clear_stammdaten,
save_stammdaten,
read_rolle,
default_rolle,
get_rolle
IMPORTING i_tx_io TYPE xubname,
set_save_button
IMPORTING i_gv_show TYPE abap_bool
i_ok_code TYPE char20,
display_rolle,

```



```

clear_rollen,
matching_rollen,
update_dyn,
set_tabs
IMPORTING i_ok_code TYPE char20,
all_rollen,
save_rollen,
empty_msg
IMPORTING i_gv_code TYPE char20.

```

PRIVATE SECTION.

* *Die Tabelle für Stammdaten*

```

DATA: BEGIN OF ms_Benutzendename,
      lt_bname      TYPE xubname, " lt_rausnehmen
      lt_Benutzendennummer TYPE usr21-persnumber,
      lt_orgnr      TYPE /lwwh/orgid,
      lt_name_first  TYPE bu_namep_f,
      lt_name_last   TYPE bu_namep_l,
      lt_smtp_addr   TYPE ad_smtpadr,
END OF ms_Benutzendename,
mt_Benutzendename LIKE TABLE OF ms_Benutzendename.

```

```

DATA: lt_agr_users TYPE TABLE OF agr_users,
      ls_agr_users TYPE agr_users.

```

```

DATA: lt_set_rollen TYPE TABLE OF agr_users,
      ls_set_rollen TYPE agr_users.

```

```

DATA: lt_dyn_agr TYPE TABLE OF /lwwh/bf_dyn_agr,
      ls_dyn_agr TYPE /lwwh/bf_dyn_agr.

```

ENDCLASS.

DATA gr_object TYPE REF TO lcl_berechtigungformula.

P01 - CLASS LCL_BERECHTIGUNGFORMULA IMPLEMENTATION.

METHOD get_stammdaten.

```

SELECT
  usr21~bname
  usr21~persnumber
  /lwwh/orgnrusr~orgnr
  adrp~name_first
  adrp~name_last
  /lwwh/orgnrusr~smtp_addr
FROM usr21
LEFT JOIN adrp ON usr21~persnumber = adrp~persnumber
LEFT JOIN /lwwh/orgnrusr ON usr21~bname = /lwwh/orgnrusr~bname
INTO TABLE mt_Benutzendename
WHERE usr21~bname = tx_io.

```

IF mt_Benutzendename IS NOT INITIAL.

```

LOOP AT mt_Benutzendename INTO ms_Benutzendename.
  tx_Benutzendenummer = ms_Benutzendename-lt_Benutzendenummer.
  IF strlen( ms_Benutzendename-lt_orgnr ) GT 3.
    tx_orgnr = ms_Benutzendename-lt_orgnr+0(3) && '.' && ms_Benutzendename-
lt_orgnr+3(3).
  ELSE.
    tx_orgnr = ms_Benutzendename-lt_orgnr.
  ENDIF.
  tx_name_first = ms_Benutzendename-lt_name_first.
  tx_name_last = ms_Benutzendename-lt_name_last.
  tx_smtp_addr = ms_Benutzendename-lt_smtp_addr.
ENDLOOP.
ELSE.
  MESSAGE: 'Zum angegebenen Schlüssel wurden keine Tabelleneinträge gefunden'
.' TYPE 'I'.
ENDIF.
IF mt_Benutzendename IS INITIAL.
  gv_show = ''.
ENDIF.

ENDMETHOD.

METHOD clear_stammdaten.

  CLEAR: tx_Benutzendenummer, tx_orgnr, tx_name_first, tx_name_last, tx_smtp_ad
dr, tx_beengrund, tx_austrittsdatum, cb_endgultig.
  IF gv_ok_code EQ 'VORLAGE'.
    CLEAR tx_io.
  ENDIF.
ENDMETHOD.

METHOD save_stammdaten.

  tx_name_first = tx_name_first.
  tx_name_last = tx_name_last.
  tx_austrittsdatum = tx_austrittsdatum.
  tx_Benutzendenummer = tx_Benutzendenummer.
  tx_beengrund = tx_beengrund.

ENDMETHOD.

METHOD set_tabs.

  IF ok_code NE 'TAB0'.

    CASE gv_ok_code.
      WHEN 'ANLEGEN'.
        gr_object->read_rolle( ).

      WHEN 'ANDERN'.
        gr_object->get_rolle( EXPORTING
i_tx_io = tx_io ).
        gr_object->display_rolle( ).
        gr_object->update_dyn( ).
    
```

```

WHEN 'ANZEIGEN'.

    gv_show = abap_false.

    gr_object->get_rolle( EXPORTING
        i_tx_io = tx_io ).

*-----*
    gr_object->display_rolle( ).

    gr_object->update_dyn( ).

WHEN 'VORLAGE'.

    gr_object->get_rolle( EXPORTING
        i_tx_io = tx_vorlage_io ).

*-----*
    gr_object->display_rolle( ).

    gr_object->update_dyn( ).

WHEN 'LOSCHEN'.
WHEN OTHERS.
ENDCASE.

ELSE.

    IF tx_io IS NOT INITIAL.
        gr_object->clear_stammdaten( ).
        gr_object->get_stammdaten(
EXPORTING
        i_tx_io = tx_io ).
    ENDIF.
ENDIF.

ENDMETHOD.

METHOD update_dyn.

    IF lt_dynpfields IS NOT INITIAL.
        CALL FUNCTION 'DYNP_UPDATE_FIELDS'
            EXPORTING
                dname      = sy-
cprog    " Program name from which the function module is called
                dynumb     = gc_dyn    " Dynpro number from ls_dynnum-
                dyn         field in current iteration
                request     = 'A'
            TABLES
                dynpfields = lt_dynpfields.    " Table containing field names and values
    ENDIF.

ENDMETHOD.

METHOD read_rolle.

```

```

CALL FUNCTION 'DYNP_VALUES_READ'
EXPORTING
  dynname          = sy-cprog    " Programmname
  dynumb           = gc_dyn      " Dynpronummer
  * translate_to_upper = space    " Großbuchstabenkonvertierung der Feldinhalt
e
  request          = 'A'         " space    " Eingabebereitschaft zurückgeben
  perform_conversion_exits = ''
  perform_input_conversion = ''
TABLES
  dynpfields       = lt_dynpfields " Tabelle zum Lesen der aktuellen Dynprowerte
EXCEPTIONS
  OTHERS           = 11.

LOOP AT lt_dynpfields INTO ls_dynpfields WHERE fieldvalue EQ 'X'.
  ls_dynpfields_save-fieldname = ls_dynpfields-fieldname.
  ls_dynpfields_save-fieldvalue = ls_dynpfields-fieldvalue.
  APPEND ls_dynpfields_save TO lt_dynpfields_save .
  CLEAR ls_dynpfields_save .
ENDLOOP.

ENDMETHOD.

METHOD save_rollen.

  LOOP AT lt_dynpfields_save INTO ls_dynpfields_save.
  *   DELETE lt_dynpfields WHERE fieldvalue = ' '.
  ENDLOOP.

ENDMETHOD.

METHOD all_rollen.

ENDMETHOD.

METHOD default_rollen.

  IF tx_Benutzendenummer IS NOT INITIAL.
    SELECT *
      FROM agr_users
      INTO TABLE lt_agr_users .

    IF sy-subrc EQ 0.
      READ TABLE lt_agr_users INTO ls_agr_users WITH KEY agr_name = 'Z:BC_SAP_
USER' .

      IF sy-subrc EQ 0.
        MOVE-CORRESPONDING ls_agr_users TO ls_set_rollen.
        ls_agr_users-uname = tx_Benutzendenummer .
        APPEND ls_set_rollen TO lt_set_rollen .
      ENDIF.

      READ TABLE lt_agr_users INTO ls_agr_users WITH KEY agr_name = 'Z:ORG_AN
ORDTYP_ANLEI' .

```

```

IF sy-subrc EQ 0.
  MOVE-CORRESPONDING ls_agr_users TO ls_set_rolle.
  ls_agr_users-uname = tx_Benutzendenummer .
  APPEND ls_set_rolle TO lt_set_rolle .
ENDIF.

READ TABLE lt_agr_users INTO ls_agr_users WITH KEY agr_name = 'Z:ORG_AN
ORDTYP_PSCD' .

IF sy-subrc EQ 0.
  MOVE-CORRESPONDING ls_agr_users TO ls_set_rolle.
  ls_agr_users-uname = tx_Benutzendenummer .
  APPEND ls_set_rolle TO lt_set_rolle .
ENDIF.

READ TABLE lt_agr_users INTO ls_agr_users WITH KEY agr_name = 'Z:DRUCKE
R_ALL' .

IF sy-subrc EQ 0.
  MOVE-CORRESPONDING ls_agr_users TO ls_set_rolle.
  ls_agr_users-uname = tx_Benutzendenummer .
  APPEND ls_set_rolle TO lt_set_rolle .
ENDIF.
ENDIF.

ENDIF.

ENDMETHOD.

METHOD get_rolle.

* IF lt_get_rolle IS INITIAL.
SELECT *
FROM agr_users
INTO TABLE lt_get_rolle
WHERE uname = i_tx_io .

IF lt_get_rolle IS INITIAL.
  MESSAGE: 'Zum angegebenen Schlüssel wurden keine Rollen gefunden.' TYPE 'I'.
ENDIF.

ENDMETHOD.

METHOD set_save_button.

gv_save_first = 'X'.

IF ( gv_ok_code EQ 'ANLEGEN' OR gv_ok_code EQ 'ANDERN' OR gv_ok_code EQ '
VORLAGE' ) AND gv_show EQ 'X'.

  SET PF-STATUS 'STATUS_0100'.

ELSEIF gv_show NE 'X' AND gv_ok_code NE 'ANLEGEN'.
  SET PF-STATUS 'STATUS_0100' EXCLUDING 'SAVE'.

```

```

ELSE.
  SET PF-STATUS 'STATUS_0100'.
ENDIF.

ENDMETHOD.

METHOD display_rollen.

  gr_object->matching_rollen().

ENDMETHOD.

METHOD matching_rollen.

  DATA:
    lv_dynpro_name TYPE char40,
    lv_index       TYPE i.

  SELECT *
    FROM /lwwh/bf_dyn_agr
    INTO TABLE lt_dyn_agr.

  SORT lt_get_rollen BY agr_name ASCENDING.
  SORT lt_dyn_agr BY agr_name ASCENDING.

  LOOP AT lt_get_rollen INTO ls_get_rollen.

    READ TABLE lt_dyn_agr INTO ls_dyn_agr WITH KEY agr_name = ls_get_rollen-
agr_name BINARY SEARCH .

    IF sy-subrc EQ 0.
      lv_index = sy-tabix .
    ENDIF.

    LOOP AT lt_dyn_agr INTO ls_dyn_agr FROM lv_index.

      IF ls_dyn_agr-agr_name NE ls_get_rollen-agr_name.
        EXIT.
      ELSE.
        CONCATENATE ls_dyn_agr-tabname ls_dyn_agr-
fieldname INTO lv_dynpro_name SEPARATED BY ' '.

        ls_dynpfields-fieldname = lv_dynpro_name .
        ls_dynpfields-fieldvalue = 'X' .

        APPEND ls_dynpfields TO lt_dynpfields.
        CLEAR ls_dynpfields .

      ENDIF.

    ENDLOOP.
  ENDLOOP.

ENDMETHOD.

```

METHOD clear_rolle.

DATA:

lv_dynpro_name **TYPE** char40,
lv_index **TYPE** i.

IF lt_dynpfields **IS NOT INITIAL**.

LOOP AT lt_dyn_agr **INTO** ls_dyn_agr.

CONCATENATE ls_dyn_agr-tabname ls_dyn_agr-
fieldname **INTO** lv_dynpro_name **SEPARATED BY** ' '.

ls_dynpfields-fieldname = lv_dynpro_name .
ls_dynpfields-fieldvalue = ' '.

APPEND ls_dynpfields **TO** lt_dynpfields.
CLEAR ls_dynpfields .

ENDLOOP.
ENDIF.

ENDMETHOD.

ENDCLASS.

PBO - /LWVH/BERECHTIGUNGFORMULA_001.

MODULE set_screen_status **OUTPUT**.

IF sy-dynnr = '100'.
CALL SCREEN 0100.

ELSEIF sy-dynnr = '101'.
CALL SCREEN 0101.

ELSEIF sy-dynnr = '110' .
CALL SCREEN 0110.

ENDIF.

IF gv_show **EQ** 'X'.

* gv_edit = space. " space = abap_false = ' '.
* gv_show = 'X'.
gv_edit = 'X'.
gv_create = 'X'.

ELSE.

* gv_show = ' '.
gv_edit = ' '.
gv_create = ' '.

ENDIF.

LOOP AT SCREEN.


```

IF screen-group4 EQ 'M01' AND gv_show NE 'X'.
    screen-input = 0.
    MODIFY SCREEN.
ELSEIF screen-group2 EQ '102'.
    screen-input = 1.
ENDIF.

IF screen-name CS 'SAVE'.
    screen-active = 0.
    MODIFY SCREEN.
ENDIF.

ENDLOOP.

ENDMODULE.

MODULE set_values OUTPUT.

ENDMODULE.

MODULE set_cursor OUTPUT.

    SET CURSOR FIELD tx_io OFFSET 1.

ENDMODULE.

MODULE status_0100 OUTPUT.

    IF gv_save_first EQ ''.
        SET PF-STATUS 'STATUS_0100' EXCLUDING 'SAVE'.
    ENDIF.

    * IF gv_show ne 'X'.
    * set PF-STATUS 'STATUS_0100' EXCLUDING 'SAVE'.
    * ENDIF.

    SET TITLEBAR 'T100'.

ENDMODULE.

MODULE status_0101 OUTPUT.
    * SET PF-STATUS 'STATUS_0101'.
    * SET TITLEBAR 'T101'.

    LOOP AT SCREEN.
        IF screen-group4 EQ 'M01' AND gv_show NE 'X'.
            screen-input = 0.
            MODIFY SCREEN.
        ENDIF.
    ENDLOOP.

ENDMODULE.

MODULE status_0110 OUTPUT.

```

```

LOOP AT SCREEN.
  IF ( /lwvh/bf_s_dynp_usr-usr_vhkanord EQ 'X' OR
    /lwvh/bf_s_dynp_usr-usr_papier EQ abap_true OR
    /lwvh/bf_s_dynp_usr-usr_technisch = 'X' ) AND
    screen-group3 EQ 'K01'.
    screen-active = 1.
    MODIFY SCREEN.
    CONTINUE.
  ELSEIF /lwvh/bf_s_dynp_usr-usr_vhkanord EQ '' AND screen-group3 EQ 'K01'.
    screen-active = 0.
    MODIFY SCREEN.
    CONTINUE.

```

```

ENDIF.

```

```

ENDLOOP.

```

```

LOOP AT SCREEN.
  IF screen-group4 EQ 'M01' AND gv_show NE 'X'.
    screen-input = 1.
    MODIFY SCREEN.
  ENDIF.
ENDLOOP.

```

```

ENDMODULE.

```

```

MODULE status_0220 OUTPUT.

```

```

LOOP AT SCREEN.
  IF screen-group4 EQ 'M01' AND gv_show NE 'X'.
    screen-input = 1.
    MODIFY SCREEN.
  ENDIF.
ENDLOOP.

```

```

ENDMODULE.

```

```

MODULE status_0600 OUTPUT.

```

```

  SET PF-STATUS '0600'.
  SET TITLEBAR 'T100'.

```

```

ENDMODULE.

```

PAI - /LWVH/BERECHTIGUNGFORMULA_I01.

```

MODULE user_command_0100 INPUT.

```

```

  IF gr_object IS NOT BOUND.
    CREATE OBJECT gr_object.
  ENDIF.

```

```

  gr_object->set_save_button(
    EXPORTING

```

```
i_gv_show = gv_show  
i_ok_code = gv_ok_code ).
```

CASE ok_code.

WHEN 'RETURN' OR 'BACK' OR 'CANCEL'.
LEAVE PROGRAM.

WHEN 'ENTER'.
gr_object->save_stammdaten().

WHEN 'SAVE'.
gr_object->save_stammdaten().
gr_object->save_rollen().

WHEN enter.
gr_object->clear_stammdaten().
gr_object->get_stammdaten(
EXPORTING
i_tx_io = tx_io).

WHEN 'ANLEGEN'.

```
gv_show = abap_true.  
gv_ok_code = 'ANLEGEN'.  
gr_object->clear_stammdaten( ).  
* gr_object->save_stammdaten( ).  
gr_object->set_save_button(
```

EXPORTING

```
i_gv_show = gv_show  
i_ok_code = gv_ok_code ).
```

```
gr_object->default_rollen( ).
```

```
gr_object->all_rollen( ).
```

WHEN 'ANDERN'.
gv_show = abap_true .
gv_ok_code = 'ANDERN'.
gr_object->clear_stammdaten().
gr_object->get_stammdaten(
EXPORTING
i_tx_io = tx_io).
gr_object->set_save_button(

EXPORTING

```
i_gv_show = gv_show  
i_ok_code = gv_ok_code ).
```

WHEN 'ANZEIGEN' .

```
gv_ok_code = 'ANZEIGEN'.  
gv_show = abap_false.
```

```
gr_object->clear_stammdaten( ).  
gr_object->get_stammdaten(
```

```

EXPORTING
i_tx_io = tx_io ).
gr_object->clear_rolle( ).

IF gc_dyn IS NOT INITIAL.
  gr_object->set_tabs( i_ok_code = gv_ok_code ).
ENDIF.

WHEN 'VORLAGE'.
  gv_ok_code = 'VORLAGE'.

CALL SCREEN 0600 STARTING AT 40 5 .
gr_object->clear_stammdaten( ).
gr_object->clear_rolle( ).
gv_show = abap_true .

IF gc_dyn IS NOT INITIAL.
  gr_object->set_tabs( i_ok_code = gv_ok_code ).
ENDIF.

WHEN 'LOSCHEN'.
  gv_ok_code = 'LOSCHEN'.
  gv_show = abap_false.
  gr_object->clear_stammdaten( ).
  gr_object->get_stammdaten(
EXPORTING
i_tx_io = tx_io ).

WHEN 'KASSENANORDNUNGEN'.

  gv_show = ''.
  IF gv_ok_code EQ 'ANZEIGEN'.
    screen-active = 0.
    MODIFY SCREEN.
  ENDIF.
WHEN 'TAB0'.

  container-activetab = 'TAB0'.
  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB1'.

  container-activetab = 'TAB1'.
  gc_dyn = gc_dyn_110 .
  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB2'.
  container-activetab = 'TAB2'.

  gc_dyn = gc_dyn_220 .
  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB2_1'.
  container-activetab = 'TAB2_1'.

```

```

gc_dyn = gc_dyn_240 .
gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB2_2'.
  container-activetab = 'TAB2_2'.
  gc_dyn = gc_dyn_241 .
  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB3'.
  container-activetab = 'TAB3'.

  gc_dyn = gc_dyn_300 .
  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB4'.
  container-activetab = 'TAB4'.
  gc_dyn = gc_dyn_400 .

  gr_object->set_tabs( i_ok_code = gv_ok_code ).

WHEN 'TAB5'.
  container-activetab = 'TAB5'.
  IF gv_ok_code eq 'VORLAGE'.
gv_ok_code = 'ANLEGEN mit VORLAGE'.
  ENDIF.

WHEN 'TX_IO'.
  IF screen-group1 EQ 110.
    screen-invisible = 0.
  ENDIF.
WHEN OTHERS.
ENDCASE.
ENDMODULE.

MODULE user_command_0101 INPUT.

ENDMODULE.

MODULE user_command_0110 INPUT.
CASE ok_code.
  WHEN 'KASSENANORDNUNGEN'.
    "CB_KASSENANORDNUNGEN = 'X'.
  WHEN ''.

  WHEN OTHERS.
ENDCASE.

ENDMODULE.

MODULE user_command_0600 INPUT.

IF gr_object IS NOT BOUND.
  CREATE OBJECT gr_object.
ENDIF.

```

```

CASE ok_code.

  WHEN 'ENTER'.
    IF tx_vorlage_io EQ ''.
      MESSAGE: 'Benutzende' TYPE 'I'.

*      CONTINUE.
    ENDIF.

    IF sy-subrc EQ 0.
      gr_object->get_rolle( i_tx_io = tx_vorlage_io ).

      IF lt_get_rolle IS INITIAL.      " Vielleicht keine Berechtigung obwohl Benutzende
gibt. TO-Do
        MESSAGE: 'Zum angegebenen Schlüssel wurden keine Tabelleneinträge gefund
en.' TYPE 'I'.
      ELSE.
        LEAVE TO SCREEN 0.
      ENDIF.

    ENDIF.
  WHEN 'BREAK'.
    LEAVE TO SCREEN 0.

  WHEN OTHERS.
ENDCASE.

CASE enter.
  WHEN 'enter'.
    IF tx_vorlage_io EQ ''.
      MESSAGE: 'Benutzende' TYPE 'I'.

    ENDIF.

    IF sy-subrc EQ 0.
      gr_object->get_rolle( i_tx_io = tx_vorlage_io ).

      IF lt_get_rolle IS INITIAL.
        MESSAGE: 'Zum angegebenen Schlüssel wurden keine Tabelleneinträge gefund
en.' TYPE 'I'.
      ELSE.
        LEAVE TO SCREEN 0.
      ENDIF.
    ENDIF.
  WHEN OTHERS.
ENDCASE.
ENDMODULE.

MODULE f4_value_request INPUT.

  DATA: lt_return  TYPE TABLE OF ddshretval,
        lt_field_tab TYPE TABLE OF dfies.

  TYPES: BEGIN OF ty_bname,
          uname TYPE xubname,

```

```

        END OF ty_bname,
        ls_bname TYPE STANDARD TABLE OF ty_bname. " WITH UNIQUE KEY uname.
DATA: lt_bname    TYPE ls_bname,
      lv_dynprofield TYPE char15.

```

**Bestimmung Name des Dynpro-Feldes für Werterückgabe*

```

IF sy-dynnr EQ '0100'.
    lv_dynprofield = 'tx_io' .
ELSE.
    lv_dynprofield = 'tx_vorlage_io' .
ENDIF.

```

** SQL Abfrage, K-nummern into lt_bname zu sammeln*

```

SELECT bname
FROM   usr01
INTO TABLE lt_bname
WHERE  bname LIKE 'K%' .

SORT lt_bname ASCENDING BY uname.

```

```

CALL FUNCTION 'F4IF_INT_TABLE_VALUE_REQUEST'
EXPORTING

```

```

    retfield      = 'BNAME'      " Name des Rückgabefeldes in FIELD_TAB
    dynpprog      = sy-repid      " Aktuelles Programm
    dynpnr        = sy-dynnr      " Dynpro-Nummer
    dynprofield   = 'lv_dynprofield' "'tx_io'      " Name des Dynpro-

```

Feldes für Werterückgabe

```

    value_org     = 'S'          " Werteübergabe: C: zellenweise, S: strukturiert

```

** callback_program = sy-repid " Programm für Callback vor F4-Start*

```

TABLES

```

```

    value_tab     = lt_bname      " Tabelle der Werte; Einträge zellenweise

```

** field_tab = lt_field_tab " Felder der Trefferliste*

```

    return_tab    = lt_return     " Rückgabe der ausgewählten Werte

```

```

EXCEPTIONS

```

```

    parameter_error = 1          " Fehlerhafte Parameter

```

```

    no_values_found = 2          " Es wurden keine Werte gefunden

```

```

    OTHERS         = 3.

```

```

ENDMODULE.

```

```

MODULE f4_date_request INPUT.

```

```

DATA: lv_dat TYPE sy-datum .

```

```

CALL FUNCTION 'F4_DATE'

```

```

EXPORTING

```

```

    gregorian_calendar_flag = 'X' "space'      " Gregorianischen Kalender anzeigen

```

```

IMPORTING

```

```

    select_date      = lv_dat      " Selektiertes Datum

```

```

EXCEPTIONS

```

```

    calendar_buffer_not_loadable = 1          " Fehler beim Lesen des Fabrikkalende

```

rs

```

    date_after_range = 2          " Datum später als Fabrikkalender

```

```

    date_before_range = 3         " Datum vor Fabrikkalender

```

```

    date_invalid      = 4         " Datum in ungültigem Format

```

```

    factory_calendar_not_found = 5          " Fabrikkalender nicht vorhanden

```

```

        holiday_calendar_not_found = 6
        parameter_conflict         = 7
geben
        OTHERS                     = 8.
        tx_austrittsdatum = lv_dat.
ENDMODULE.
MODULE user_command_0500 INPUT.
IF gv_ok_code eq 'VORLAGE'.
gv_ok_code = 'ANLEGEN mit VORLAGE'.
ENDIF.
ENDMODULE.

```

" Feiertagskalender nicht vorhanden
" Fabrikkalender und Feiertagskalender über

Erstellen eines kurzen Konzepts zur folgenden Problemstellung: (Org_ID)

- Einführung

Fachbereich 102.0 möchte einen „Programm“. Auf dem Bildschirm soll es zwei Eingabefelder geben. In einem Eingabefeld soll man Datum eintragen können und in dem anderen sollen Status eintragen können. Außerdem gibt es auch eine Checkbox, mit der Checkbox die Ergebnisse als Gruppe angezeigt werden. Nachdem erfolgreichen Eingaben soll User Ergebnis „Durchführen“ Button anklicken.

Anschließend soll es eine Ausgabe angezeigt werden. Darauf soll eine Tabelle, auf der die Org-ID und die Anzahl aufgeführt sind, wie folgendes Beispiel dargestellt werden.

Org-ID	Anzahl

- Ist-Stand

Zurzeit gibt es keine Dasselbe oder Ähnliches.

- Projektziel

Dieses Konzept bietet ein Konzept, das angezeigt wird, wie viele Rechnungen je nach Org-ID erstellt werden.

In 30 Tagen wird das Programm umgestellt werden. Das Programm soll Admin helfen, in kürze Zeit eine bestimmte Eingabe, Datum und Status, je nach Fachbereich oder Org_ID auflisten zu lassen und produktiv zu arbeiten.

Das Programm soll die entsprechende Tabelle wird als ALV-GRID erstellt. Damit können die Ergebnisse visuell Schöneres aufgelistet werden.

In den ersten 20 Tagen wird Kodierung fertiggestellt und geht das Programm als Test Version. Dieser Zeitraum wird es als Entwurf genannt und getestet.

- Beschreibung des Geschäftsprozess

Das Programm hilft dabei, dass Firma damit Zeit sparen kann.

Die aufgeführten Ergebnisse können die Vertraulichkeit des Unternehmens verletzen. Zu

eine
Das

diesem
Zweck ist
Autorisierung
vorgesehen.
Programm ist
vorgesehen,

dass es von nur Admin benutzt wird.

- . Anforderungen an die Funktionalität

5.1. Benutzeroberfläche:

1.1. Erstellen eine Benutzeroberfläche mit zwei Eingabefeldern, eine Checkbox und einem Anzeige-Button. Checkbox für die Gruppierung der Ergebnisse.

1.2 Das erste Eingabefeld ist für Datum und das zweite Eingabefeld für Status, dazwischen eine Checkbox, wie im folgende dargestellt werden.

1.2.1 Eingabefeld für Datum ist kein Wert angezeigt. Sofern User Felder für Datum leer lässt, zeigt das Programm eine Information, welche das Ergebnis mit viele Einträge angezeigt werden.

1.2.2 Eingabefeld für Status ist mit Default Values angezeigt, welche von 1 bis 4. Sofern User noch bis 5 auflisten lassen möchte, dann zeigt Programm eine Information, welche das Ergebnis als gelöschten gekennzeichneten Einträgen angezeigt werden.

1.2.3 Checkbox ist als leer, d.h. ohne Auswahl, angezeigt werden.

Ausgabe: Gruppiert nach Fachbereich

Org-ID	Anzahl
102	12
103	21
103	24
104	14

Ausgabe Gruppiert nach Org-Nummer

Org-ID	Anzahl
102	2
1020	3
1020005	8
1020006	12

- Technische Lösung

Anzahl: Anzahl zeigt die Gesamtzahl einer Org-ID.

Status: Status wird als **Status** in Tabelle/Datenbank gespeichert.

Datum: Datum wird als **EF_Datum** in Tabelle/Datenbank gespeichert.

Database: Der Name der Tabelle ist **/LWVH/RS_RCHNK**.

Der Name des Programms: **/LWVH/RS_RCHNK_Auswert** – Anzahl der OrgID

Dafür hat es in 4 Schritte dargestellt. Erste ist es über Darstellung einer Benutzeroberfläche. Dann kommt es zweite Schritt als Validierung der Eingaben, d.h. die Eingabe überprüft, ob die Felder richtig gefüllt sind. Anschließend wird Eingabe mit Hilfe einen Algorithmus von Server abgerufen (Schritt-3) und auf eine Tabelle angezeigt (Schritt-4).

6.1. Benutzeroberfläche:

Auf Kapitel 5 ist es schon genannt werden, daher ist es hier nicht noch einmal eingeschrieben.

6.2. Validierung der Eingaben:

2.4 Wenn beide Felder gefüllt sind, wird die Validierung fortgesetzt (Automatisch SAP-Prüfung).

6.3. Durchführung

3.1. Wenn das Datum und Status Felder richtig gefüllt sind (Automatisch SAP-Prüfung), wird die Ergebnis je nach Org_ID aufgelistet werden.

3.2. Wenn das Datum und Status Felder richtig gefüllt sind und Checkbox angekreuzt ist, wird die Ergebnis je nach Fachbereich aufgelistet werden.

3.3 Selektieren entsprechende Tabelle.

Tabelle ist **/LWVH/RS_RCHNK**

In diesem Beispiel wird angenommen, dass Database mit SQL-Befehl ausgewählt wird. Nach der Selektion des Database kann auch die entsprechende SQL Query durchgeführt werden.

3.4 Um die Fachbereiche in SQL zu gruppieren, können Sie die Funktion GROUP BY verwenden.

Mithilfe des **GROUP BY** wird definiert, wie die Datenmenge gruppiert werden soll. Die Ergebnismenge kann nach mehreren Spalten gruppiert werden. Hier ist zwei Beispiel für eine SQL-Abfrage, die die Org-ID und deren Anzahl pro Fachbereich zurückgibt:

3.3.1 Wenn Checkbox nicht angekreuzt ist:

```
DATA: lt_data TYPE TABLE OF /LWVH/RS_RCHNK,  
      ls_data TYPE /LWVH/RS_RCHNK.
```

```
SELECT anzahl, COUNT(*) AS anzahl, ef_organ AS ef_organ  
      INTO TABLE lt_data  
      FROM /LWVH/RS_RCHNK  
      GROUP BY ef_organ.
```

Write: ' ORG_ID ', 10 'Anzahl' .

LOOP AT lt_data INTO ls_data.

WRITE: / ls_data-anzahl under 'Anzahl', ls_data-ef_organ under ' ORG_ID '.

ENDLOOP.

In diesem Beispiel wird eine interne Tabelle lt_data vom Typ /LWVH/RS_RCHNK definiert und eine Work-Area ls_data vom gleichen Typ. Die SELECT-Anweisung wird verwendet, um Daten aus der Tabelle lt_data abzurufen und in die interne Tabelle zu laden. Die Spalten "Anzahl", COUNT(*) (als anzahl umbenannt) und "EF_ORGID" werden ausgewählt und nach der EF_ORGID gruppiert.

Dann wird mit einer Schleife über die interne Tabelle iteriert und jeder Datensatz ausgegeben.

3.3.2 Wenn Checkbox angekreuzt ist:

```
DATA: lt_data TYPE TABLE OF /LWVH/RS_RCHNK,  
      ls_data TYPE /LWVH/KVH/RS-RCHNK.
```

```
SELECT anzahl, COUNT(*) AS anzahl, ef_orgid, 3 AS ef_orgid  
      INTO TABLE lt_data  
      FROM /LWVH/RS_RCHNK  
      GROUP BY ef_orgid, 3 .
```

```
Write: ' ORG_ID ' , 10 'Anzahl' .
```

```
LOOP AT lt_data INTO ls_data.  
  WRITE: / ls_data-anzahl under 'Anzahl', ls_data-ef_orgid under ' ORG_ID ' .  
ENDLOOP.
```

In diesem Beispiel wird eine interne Tabelle lt_data vom Typ /LWVH/RS-RCHNK definiert und eine Work-Area ls_data vom gleichen Typ. Die SELECT-Anweisung wird verwendet, um Daten aus der Tabelle /LWVH/RS_RCHNK abzurufen und in die interne Tabelle zu laden. Die Spalten "Anzahl", COUNT(*) (als anzahl umbenannt) und "EF_ORGID" werden ausgewählt und nach der EF_ORGID mit ersten Ziffern gruppiert.

Anschließend wird eine Schleife verwendet, um über die interne Tabelle zu iterieren und jeden Datensatz auszugeben.

6.4. Anzeige der Ergebnisse:

4.1 Nach erfolgreicher Durchführung wird das Ergebnis auf dem Ausgabebereich angezeigt.

Das Ergebnis dieser Abfrage wäre eine Tabelle mit zwei Spalten: Fachbereich „ORG_ID“ und „Anzahl“. Jede Zeile repräsentiert eine ORG_ID und zeigt dessen entsprechende Org-ID-Anzahl an.

Org-ID	Anzahl
102	2
1020	3
1020005	8
1020006	12

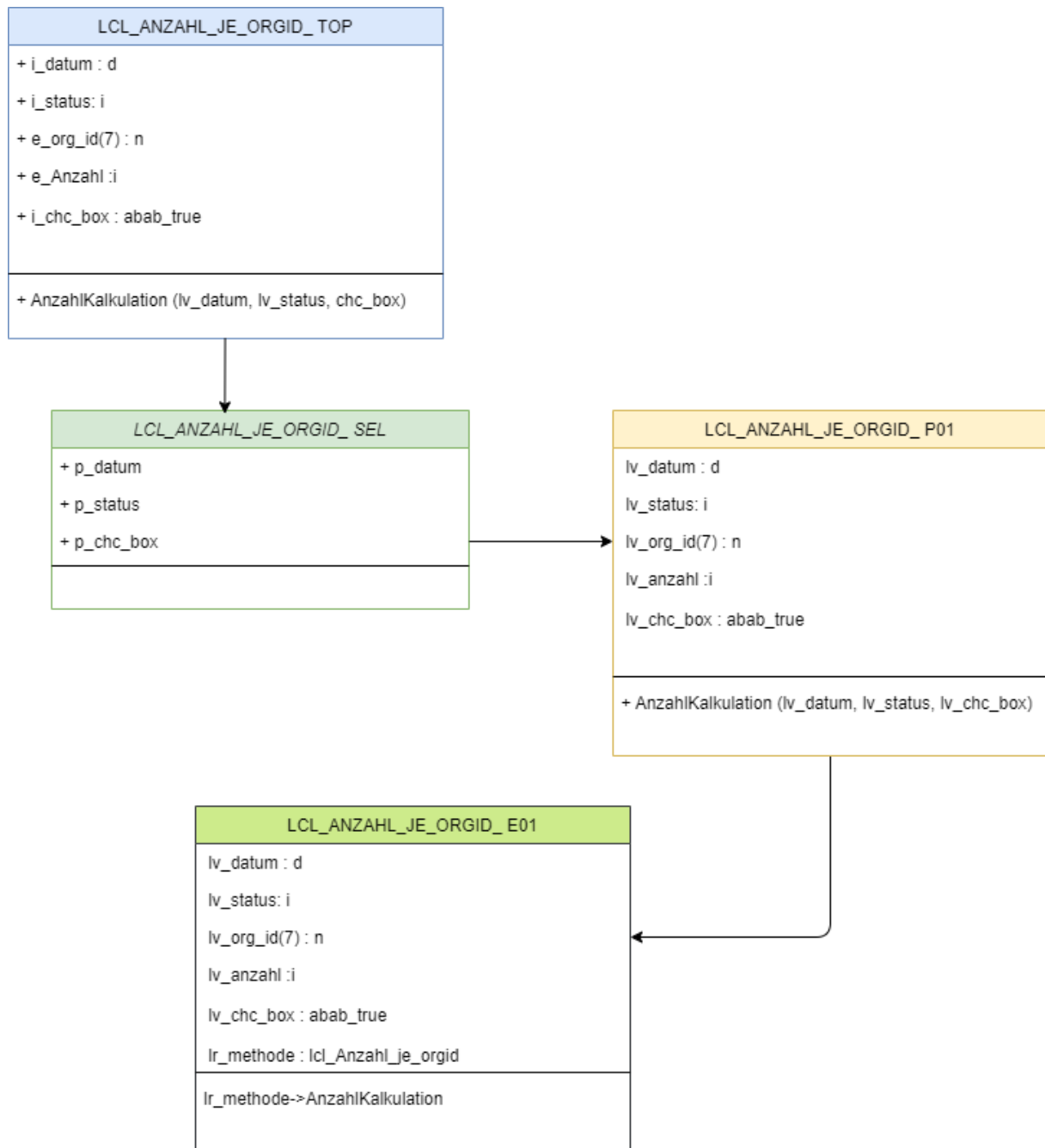
4.2 Nach erfolgreicher Durchführung wird das Ergebnis mit Checkbox Option auf dem Ausgabebereich angezeigt. Die Ergebnisse werden nach der Gesamtzahl der Fachbereiche angezeigt.

Das Ergebnis dieser Abfrage wäre eine Tabelle mit zwei Spalten: Fachbereich „ORG_ID“ und „Anzahl“. Jede Zeile repräsentiert einen Fachbereich und zeigt dessen entsprechende Org-ID-Anzahl an.

Org-ID	Anzahl
102	12
103	21
103	24
104	14

6.5. Implementierungsdetails:

5.1 Verwenden von geeigneten Funktionen oder Algorithmen zur Durchführung den Eingaben und Validierungen wie folgenden.





Report /LWVH/RS_RCHNK_AUSWERT aktiv

```
1  *-----*
2  *
3  * P R O G R A M M - I N F O R M A T I O N
4  *-----*
5  *Entwickler:      | Sefa Alioglu (K01508)
6  *Abteilung:      | LWVH (FB102)
7  *Telefon:        | +(49)561 1004 / 2731
8  *E-Mail:         | SefaPrakt.Alioglu@lwv-hessen.de
9  *Programmname    | /LWVH/RS_RCHNK_AUSWERT
10 *Datum:          | 08.11.2023 12:12:24
11 *-----*
12 *Programmbeschreibung:
13 * SAP23-089: Auswertung zum Faktura-Nutzungsgrad
14 *
15 *-----*
16 * Änderungsbeschreibungen
17 *-----*
18 REPORT /lwvh/rs_rchnk_auswert.
19
20 INCLUDE /lwvh/rs_rchnk_auswert_top.
21 INCLUDE /lwvh/rs_rchnk_auswert_sel.
22 INCLUDE /lwvh/rs_rchnk_auswert_e01.
23 INCLUDE /lwvh/rs_rchnk_auswert_p01.
```

Programm zum Faktura-Nutzungsgrad



Status und Datum

Status	<input type="text" value="1"/>	bis	<input type="text" value="4"/>	
Datum	<input type="text"/>	bis	<input type="text"/>	

Gruppierung der Ergebnisse

☒ Gruppierung

Konzept zur Erstellung von CSV-Reports für Vitos

Überblick

Die IT-Abteilung der Vitos hat eine Anfrage zur Programmierung zweier Reports gestellt. Diese Reports sollen Daten aus verschiedenen SAP-Tabellen extrahieren und in zwei .csv-Dateien exportieren. Die .csv-Dateien sollen gemäß den Beispieldateien formatiert sein und auf dem SAP-Applikationsserver im Verzeichnis /saplww/E10/trans/DMS/ gespeichert werden.

Anforderungen

Report 1: Innenaufträge

Programmname: /LWVH/VITOS_AUFGK

SAP-Tabelle: AUFGK

Dateiname: SAP_AUFGK.csv

Kriterien:

Berücksichtigt werden nur Innenaufträge des Kostenrechnungskreises 1000.

Möglichkeit zur Einschränkung der Buchungskreise durch eine Eingabemaske (Von-Buchungskreis und Bis-Buchungskreis).

SAP_AUFGK.csv =Dateiname				
AUFGNR	KTEXT	AUTYP (Type)	BUKRS	SAP Feldname
ID	TEXT	TYPE	ACCOUNTING_AREA	CSV-Datei Feldname

Feldtrennzeichen: Komma

Zeilenende: CR-LF (Windows-Standard)

Eingabemaske:

Von-Buchungskreis

Bis-Buchungskreis

Implementierungsdetails

Report 1: /LWVH/VITOS_AUFGK

Felder für „Von-Buchungskreis“ und „Bis-Buchungskreis“ hinzufügen.

Datenextraktion

Daten aus Tabelle AUFK basierend auf den Kriterien KOKRS = 1000 und Buchungskreisfilter extrahieren.

CSV-Erstellung

Exportierte Daten in das Format: AUFNR, KTEXT, AUTYP (Type), BUKRS konvertieren.

Daten mit Komma trennen und Zeilen mit CR-LF abschließen.

Speicherung

Die generierte .csv-Datei unter /saplww/E10/trans/DMS/ als SAP_AUFK.datum.csv speichern.

Zusätzliche Hinweise

Testen:

Vor der endgültigen Bereitstellung sollten beide Reports gründlich getestet werden, um sicherzustellen, dass die Kriterien korrekt angewendet werden und die .csv-Dateien das korrekte Format haben.

Notepad-Überprüfung:

Um sicherzustellen, dass die .csv-Dateien korrekt formatiert sind, sollten sie mit Notepad geöffnet und überprüft werden.

Zugriff auf Verzeichnisse:

Verzeichnisse des SAP-Applikationsservers können über den Report RN2LN205N (durch SE38) eingesehen und Dateien auf lokale Laufwerke heruntergeladen werden.

Abschluss

Durch die Implementierung dieser Reports wird sichergestellt, dass die IT-Abteilung der Vitos die benötigten Daten in einem standardisierten Format erhält, das leicht weiterverarbeitet werden kann. Die genaue Einhaltung der Anforderungen und eine gründliche Testphase sind entscheidend für den Erfolg dieses Projekts. Das Projekt ist erfolgreich implementiert und zu dem Mitarbeitenden zur Verfügung gestellt.

Report 2: Bankverbindungen der Lieferanten (BNKA)

Überblick

Die IT-Abteilung der Vitos hat eine Anfrage zur Programmierung zweier Reports gestellt. Diese Reports sollen Daten aus verschiedenen SAP-Tabellen extrahieren und in zwei .csv-Dateien exportieren. Die .csv-Dateien sollen gemäß den Beispieldateien formatiert sein und auf dem SAP-Applikationsserver im Verzeichnis /saplww/E10/trans/DMS/ gespeichert werden.

Programmname: /LWVH/VITOS_BNKA

SAP-Tabellen: LFBK, BNKA (Name der Bank), TIBAN (IBAN)

Dateiname: SAP_BNKA.csv

Kriterien:

Berücksichtigt werden nur Lieferantennummern von ,0007000000' bis ,0007999999'.

SAP_BNKA.csv =Dateiname							
LIFNR	BANKA	BANKL	BANKN	IBAN	SWIFT	BVTYP (LFBK)	SAP Feldname
CREDITOR	BANK	CODE	ACCOUNT	IBAN	BIC	PBT	CSV-Datei Feldname

Feldtrennzeichen: Komma

Zeilenende: CR-LF (Windows-Standard)

Report 2: /LWVH/VITOS_BNKA

Datenextraktion

Daten aus Tabellen LFBK, BNKA und TIBAN basierend auf dem Kriterium LIFNR zwischen ,0007000000' und ,0007999999' extrahieren.

Datenverknüpfung

Verknüpfung der relevanten Felder: LIFNR, BANKA, BANKL, BANKN, IBAN, SWIFT, BVTYP.

CSV-Erstellung

Exportierte Daten in das Format: LIFNR, BANKA, BANKL, BANKN, IBAN, SWIFT, BVTYP konvertieren.

Daten mit Komma trennen und Zeilen mit CR-LF abschließen.

Speicherung

Die generierte .csv-Datei unter /saplwv/E10/trans/DMS/ als SAP_BNKA.datum.csv speichern.

Zusätzliche Hinweise

Testen:

Vor der endgültigen Bereitstellung sollten beide Reports gründlich getestet werden, um sicherzustellen, dass die Kriterien korrekt angewendet werden und die .csv-Dateien das korrekte Format haben.

Notepad-Überprüfung:

Um sicherzustellen, dass die .csv-Dateien korrekt formatiert sind, sollten sie mit Notepad geöffnet und überprüft werden.

Zugriff auf Verzeichnisse:

Verzeichnisse des SAP-Applikationsservers können über den Report RN2LN205N (durch SE38) eingesehen und Dateien auf lokale Laufwerke heruntergeladen werden.

Abschluss

Durch die Implementierung dieser Reports wird sichergestellt, dass die IT-Abteilung der Vitos die benötigten Daten in einem standardisierten Format erhält, das leicht weiterverarbeitet werden kann. Die genaue Einhaltung der Anforderungen und eine gründliche Testphase sind entscheidend für den Erfolg dieses Projekts. Das Projekt ist erfolgreich implementiert und zu dem Mitarbeitenden zur Verfügung gestellt.

Erstellen eines kurzen Konzepts zur folgenden Problemstellung: (DEB1 und DEB2)

- Einführung

Der Fachbereich 102.0 möchte zwei „Programme“ nämlich /LWVH/ZFI_FAELLIGKEITEN_DEB1 und /LWVH/ZFI_FAELLIGKEITEN_DEB2 anpassen lassen. Die Programme prüfen, ob der Benutzer Zugriff auf alle Vitos Buchungskreise Berechtigungen hat, dann laufen die entsprechenden Programme. Wenn der Benutzer nicht alle Zugriffs-Rechte hat, laufen die Programme nicht. Auf dem Bildschirm soll es ein Eingabefeld geben. In einem Eingabefeld soll man Buchungskreise eintragen können.

- Ist-Stand

Zurzeit gibt es die Programme aber es wird dafür Zugriff auf die Berechtigungen aller Vitos Buchungskreise benötigt. Bisher wurden die Programme nur von FB 102.0 genutzt.

- Soll-Stand

Jetzt sollen die Gesellschaften die Programme nutzen. Je nach Zugriffsberechtigung auf die Vitos Buchungskreise Berechtigungen laufen die Programme, wenn die Vitos Buchungskreise gültig sind und Ergebnisse erfüllt werden.

- Projektziel

Das Programm läuft nur, sofern der Benutzer Zugriff auf die entsprechende Buchungskreise hat. Dann laufen die Programme mit den Ergebnisdaten der berechtigten Buchungskreise.

- Beschreibung des Geschäftsprozess

Das Programm hilft dabei, dass der Benutzer das Programm nur starten kann, wenn er für die BUK (Buchungskreise) auch die Berechtigungen hat.

- . Anforderungen an die Funktionalität

6.1. Benutzeroberfläche:

6.1.1. Erstellen eines Eingabefeldes, in dem Buchungskreise auswählen können.

- Technische Lösung

Die Namen der Programme: /LWVH/ZFI_FAELLIGKEITEN_DEB1 und /LWVH/ZFI_FAELLIGKEITEN_DEB2.

Es wird in 3 Schritte implementiert. Der Erste ist die Darstellung einer Benutzeroberfläche. Dann kommt der zweite Schritt zur Validierung der Eingaben, d.h. die Eingabe wird überprüft, ob der Zugriff, bzw. der Inhalt richtig gefüllt ist. Anschließend wird die Eingabe mit Hilfe eines Algorithmus von Server abgerufen.

7.1. Benutzeroberfläche:

Auf Kapitel 6 ist es schon genannt werden, daher ist es hier nicht noch einmal eingeschrieben.

7.2. Validierung der Eingaben:

Wenn beide Felder gefüllt sind, wird die Validierung fortgesetzt (Automatisch SAP-Berechtigungsprüfung).

7.3. Durchführung

7.3.1. Wenn die Buchungskreise richtig gefüllt sind (Automatisch SAP-Prüfung), wird das Ergebnis aufgelistet werden.

In diesem Beispiel wird angenommen, dass die Database mit einem SQL-Befehl ausgewählt wird. Nach der Selektion des Database kann auch die entsprechende SQL Query durchgeführt werden.

Mithilfe des SQL-Befehls **IN** wird implementiert, wie die Datenmenge aufgelistet werden soll.

7.4. Anzeige der Ergebnisse:

Wie schon früher aufgelistet und angezeigt.

7.5. Implementierungsdetails:

7.5.1 Verwenden von geeigneten Funktionen oder Algorithmen zur Durchführung den Eingaben und Validierungen wie die folgenden.

7.5.1.1 Deklaration.

Dafür wurde eine ITAB (mt_bukrs) deklariert. Damit werden die alle gültige und erhaltene Buchungskreise den Zugriffsrechten gesammelt.

Eine andere ITAB wurde deklariert, nämlich lt_buk_zu. Hier werden die entsprechenden Buchungskreise von Tabelle T001, Werks von Tabelle T001w, Bewertungskreise von Tabelle T001k und die Namen der Buchungskreise von Tabelle T001 erfasst.

Drei Attributen von der Tabelle intabkh sind je nach angepassten Datentype laut entsprechende Database Tabellen, nämlich T001w-werks, T001K-bwkey, T001-butxt, korrigiert. Früher waren sie vom type C.

Ein Abap_bool nämlich lv_flag wurde auch deklariert, zudem die Struktur des Programmes laufen zu lassen. Früher wurde es auf alle Berechtigungen geprüft danach ist es damit gelaufen. Um die Struktur des Programmes zu behalten, wird dieses Attribut benutzt, Sofern ein Benutzer eine oder mehrere Zugriffsrechte hat.

Authority Check

Anstatt eines statischen Authority Check wurde ein dynamischer Authority Check mit Hilfe Loop At mit der Database Tabelle T001 implementiert. Diese Ergebnisse werden die oben genannte Tabelle lt_bukrs gesammelt. Authority Check wurde in der Method „start_of_selection“ implementiert.

7.5.1.3 Pflege und Korrigieren der Methoden

7.5.1.3.1 Method: KHAUS_INIT.

Als erstes wurden drei itab deklariert, indem von drei Database Tabellen von T001, T001k, T001w die entsprechenden Informationen abgerufen werden.

Tabelle It_t001 ist für die Gültigkeit von Buchungskreise.
Tabelle It_t001k ist für die Bewertungskreise je nach dem Buchungskreise.
Danach werden alle Informationen werden in der Tabelle It_buk_zu gesammelt.
Die Zuordnung der Buchungskreise zu Gesellschaften in dieser Tabelle wird in der Ausgabe verwendet.

7.5.1.3.2 Method: BSID_LESEN

Mit Hilfe Loop At von It_bukrs, welche erhaltene Zugriffsrechte schon erfasst, lassen sich die entsprechende Buchungskreise einschränken. Alle Änderungen wurde mit einer Versionswarnung 1.X angezeigt.

7.5.1.3.3 Method: BRSCH_SUMME

Die Wertzuweisung ist zuvor mit Offset erfolgt. Jetzt mit Hilfe Loop von It_buk_zu, die mit zuvor gesammelten Informationen erfasst werden.

Die Befehle READ TABLE intabkh WITH KEY k_sort = p_ges INTO ksatz und MODIFY intabkh FROM ksatz TRANSPORTING dmbtr0 dmbtr1 dmbtr2 dmbtr3 dmbtr4 dmbtr5 dmbtr6 dmbtr7 dmbtr8 dmbtr9 dmbtr10 dmbtr11 dmbtr12 WHERE k_sort = p_ges wurden gepflegt.

7.5.1.3.4 Method: ALV_ANZEIGEN

Die Ergebnisse der Tabelle intabkh wird mit BY k_sort sortiert. Die Ergebnisse werden je nach dem Buchungskreis angezeigt.

7.5.1.3.5 Method: BUK_SETZEN

Die entsprechende Wertzuweisungen wurde mit Hilfe loop at It_buk_zu erledigt.

7.5.1.3.6 Method: ANZEIGE_BUK

Zuordnung ist rausgezogen.

7.5.1.3.7 Method: start_of_selection

Wurde als neu method definiert, um start of selection, zu verkürzen und zu vereinfachen.

Authority Check und range_table wurden hier implementiert.

Es wurde eine Tabelle mit Range option, so_bukrs definiert. Damit wird beim Select in bsid_lesen die Performance verbessert.

Erstellen eines kurzen Konzepts zur folgenden Problemstellung: (Taschenrechner)

Einführung

Fachbereich 102.0 möchte einen „Programm“. Es soll ein Taschenrechner wie bereits in Windows ist. Es soll aber eine freie Eingabe sein, welche nacheinander Rechenaufgaben einzugeben. Es soll ja Historie anzeigen, welche Benutzer bisher eingegeben hat.

Ist-Stand

Zurzeit gibt es keine Dasselbe oder Ähnliches.

Projektziel

Dieses Konzept bietet ein Konzept, das angezeigt wird, wie die Rechnung Aufgabe ist.

Beschreibung des Geschäftsprozess

Das Programm hilft dabei, dass ABAP-Kenntnisse verbessert werden kann.

. Anforderungen an die Funktionalität

Benutzeroberfläche:

- 1.1. Erstellen eine Benutzeroberfläche mit Eingabefeldern, wie bereits im Win PC.

Technische Lösung

Benutzeroberfläche:

Wie im Folgenden.

SAP

Ergebnis

MC	MR	MS	M+	M-
< RETURN	CE	C	+-	✓ WURZEL
7	8	9	/	%
4	5	6	*	1/x
1	2	3	-	=
0	,	+	EINGABE	<input type="checkbox"/> FREI

Validierung der Eingaben:

Mit Hilfe eine Methode kann es geprüft werden, ob es Valid ist.

Anzeige der Ergebnisse:

SAP

Ergebnis

MC	MR	MS	M+	M-
< RETURN	CE	C	+-	✓ WURZEL
7	8	9	/	%
4	5	6	*	1/x
1	2	3	-	=
0	,	+	EINGABE	<input type="checkbox"/> FREI

Ergebnis

3*9+6+/2+3+9+4+2-8*8/3

MC	MR	MS	M+	M-
< RETURN	CE	C	+ -	✓ WURZEL
7	8	9	/	%
4	5	6	*	1/x
1	2	3	-	=
0	,	+	EINGABE	<input checked="" type="checkbox"/> FREI

! Eingegebenes Formula ist ungültig

Ergebnis

110

3*6*8*9/6/2+3-1

MC	MR	MS	M+	M-
< RETURN	CE	C	+ -	✓ WURZEL
7	8	9	/	%
4	5	6	*	1/x
1	2	3	-	=
0	,	+	EINGABE	<input checked="" type="checkbox"/> FREI

SAP

Ergebnis

2.33

3*6/5/4+2*5/7

MC	MR	MS	M+	M-
< RETURN	CE	C	+-	✓ WURZEL
7	8	9	/	%
4	5	6	*	1/x
1	2	3	-	=
0	,	+	EINGABE	<input checked="" type="checkbox"/> FREI

6.5. Implementierungsdetails:

Verwenden von geeigneten Funktionen oder Algorithmen zur Durchführung den Eingaben und Validierungen wie folgenden.

&-----

*& Include /LWVH/TASCHENRECHNERDYNPRO_TOP
/LWVH/TASCHENRECHNERDYNPRO

Modulpool

*&

&-----

PROGRAM /LWVH/TASCHENRECHNERDYNPRO.

*CONSTANTS:

```
data: ok_code TYPE sy-ucomm ,
      enter TYPE sy-ucomm,
      tx_io TYPE char20,
      tx_eingabe TYPE char20,
      tx_temp TYPE char20,
      gv_optemp TYPE char1,
```

gv_temp_zahl1 TYPE char20,
tb_mc TYPE char2,
tb_mr TYPE char2,
tb_ms TYPE char2,
tb_mplus TYPE char2,
tb_mminus TYPE char2,
tb_return TYPE char2,
tb_ce TYPE char2,
tb_c TYPE char2,
tb_plmn TYPE char2,
tb_zero TYPE char1,
tb_eins type char1,
tb_zwei TYPE char1,
tb_drei TYPE char1,
tb_vier TYPE char1,
tb_funf TYPE char1,
tb_sechs TYPE char1,
tb_sieben TYPE char1,
tb_acht TYPE char1,
tb_neun TYPE char1,
tb_plus TYPE char1,
tb_mal TYPE char1,
tb_div TYPE char1,
tb_minus TYPE char1,
tb_komma TYPE char1,
tb_prozent TYPE char1,
tb_wurzel TYPE char1,
tb_gleich TYPE char1,

tb_x1 TYPE char1,
gv_ergebnis TYPE int8,
gv_ergebtemp TYPE int8,
gv_tempe TYPE int8,
gv_tempd TYPE p DECIMALS 2,
gv_decimal TYPE p DECIMALS 2,
gv_decitemp TYPE p DECIMALS 2,
gv_kommaaktiv TYPE abap_bool,
gv_auswahl TYPE char7,
gv_counter TYPE i,
gv_flag_nummer TYPE boolean VALUE 'X',
cb_frei type xfeld.

CLASS lcl_taschenrechner DEFINITION.

PUBLIC SECTION.

METHODS: constructor,

* plus,
* minus,
* mal,
* div,
* wurzel,
* mc,
* mr,
* ms,
* mplus,
* mminus,
* prozent,
* reciprocal,

anzeigen_temp,
haupt,
eingabe,
reset,
c_ce.

PRIVATE SECTION.

METHODs:

plus,
minus,
mal,
div,
wurzel,
mc,
mr,
ms,
mplus,
mminus,
prozent,
reciprocal.

* anzeigen_temp,
* haupt.
* eingabe,
* reset,
* c_ce.

ENDCLASS.

&-----

*& Include /LWVH/TASCHENRECHNERDYNPRO_O01

&-----

MODULE status_0111 OUTPUT.

SET PF-STATUS 'STATUS_0111'.

SET TITLEBAR 'T111'.

LOOP AT SCREEN.

* IF cb_frei EQ 'X' AND (screen-group1 EQ 'L1' OR screen-group1 EQ 'L2' OR screen-group1 EQ 'NUM' OR screen-group1 EQ 'OPR') .

IF cb_frei EQ 'X' AND (screen-group1 ne 'FRI' and screen-group1 ne 'SHW' and screen-group1 ne 'IO1' and screen-group2 ne 'L2E').

screen-input = 0.

MODIFY SCREEN.

ENDIF.

ENDLOOP.

ENDMODULE.

&-----

*& Include /LWVH/TASCHENRECHNERDYNPRO_P01

&-----

CLASS lcl_taschenrechner IMPLEMENTATION.

METHOD constructor.

ENDMETHOD.

METHOD haupt.

CASE gv_auswahl.

WHEN 'PLUS'.

me->plus().

WHEN 'MINUS'.

me->minus().

WHEN 'MAL'.

me->mal().

WHEN 'DIV'.

me->div().

WHEN 'WURZEL'.

me->wurzel().

ENDCASE.

IF ok_code EQ 'X'.

me->reciprocal().

ENDIF.

ENDMETHOD.

METHOD plus.

IF gv_counter EQ 1 .

gv_tempe = tx_io .

gv_tempd = tx_io .

ELSE.

gv_tempe = tx_io + gv_tempe .

gv_tempd = tx_io + gv_tempd.

ENDIF.

gv_ergebnis = gv_tempe . "+" tx_io. "tx_io +

gv_decimal = gv_tempd . "+" tx_io .

CLEAR tx_io.

ENDMETHOD.

METHOD minus.

IF gv_counter EQ 1.

gv_tempe = tx_io .

gv_tempd = tx_io .

ELSE.

gv_tempe = gv_tempe - tx_io .

gv_tempd = gv_tempd - tx_io .

ENDIF.

gv_ergebnis = gv_tempe .

gv_decimal = gv_tempd .

CLEAR tx_io.

ENDMETHOD.

METHOD mal.

TRY.


```

IF gv_counter EQ 1.
    gv_tempe = tx_io .
    gv_tempd = tx_io .
ELSE.
    gv_tempe = gv_tempe * tx_io .
    gv_tempd = gv_tempd * tx_io .
ENDIF.

CATCH cx_sy_arithmetic_overflow.

ENDTRY.

gv_ergebnis = gv_tempe .
gv_decimal = gv_tempd .
CLEAR tx_io .

ENDMETHOD.

METHOD div.
    IF tx_io EQ 0.
        MESSAGE: 'Teilen durch 0 nicht möglich' TYPE 'E'.
    ELSEIF gv_counter EQ 1.
        gv_tempe = tx_io .
        gv_tempd = tx_io .
    ELSE.
        gv_tempe = gv_tempe / tx_io .
        gv_tempd = gv_tempd / tx_io .
    ENDIF.

```

```

gv_ergebnis = gv_tempe .
gv_decimal = gv_tempd .
CLEAR tx_io .
ENDMETHOD.

```

METHOD wurzel.

```

DATA: lv_temp TYPE p DECIMALS 4.
lv_temp = ( 1 / tx_io ) .
gv_ergebnis = gv_ergebnis ** lv_temp .
gv_decimal = gv_decimal ** lv_temp .
ENDMETHOD.

```

METHOD prozent.

CASE gv_auswahl.

WHEN 'PLUS'.

```

gv_decimal = gv_decimal - tx_io .
gv_decimal = ( ( tx_io * gv_decimal ) / 100 ) + gv_decimal .

```

WHEN 'MINUS'.

```

gv_decimal = gv_decimal + tx_io .
gv_decimal = gv_decimal - ( ( tx_io * gv_decimal ) / 100 ) .

```

WHEN 'MAL'.

```

gv_decimal = gv_decimal / tx_io .
gv_decimal = ( tx_io * gv_decimal ) / 100 .

```

WHEN 'DIV'.

```

gv_decimal = gv_decimal * tx_io .
gv_decimal = ( 100 * gv_decimal ) / tx_io .

```

WHEN OTHERS.

ENDCASE.

* ENDIF.

ENDMETHOD.

METHOD mc.

* CLEAR: gv_ergebnis, gv_decimal.

ENDMETHOD.

METHOD mr.

tx_io = gv_ergebtemp .

tx_io = gv_decitemp .

CASE gv_auswahl.

WHEN 'PLUS'.

mplus().

WHEN 'MINUS'.

mminus().

WHEN OTHERS.

ENDCASE.

ENDMETHOD.

METHOD ms.

IF gv_auswahl EQ 'MPLUS'.

me->mplus().

ELSEIF gv_auswahl EQ 'MMINUS'.

me->mminus().

ENDIF.

ENDMETHOD.

METHOD mplus.

gv_ergebtemp = gv_ergebnis + tx_io .

gv_decitemp = gv_decimal + tx_io .

ENDMETHOD.

METHOD mminus.

gv_ergebtemp = gv_ergebtemp - tx_io .

gv_decitemp = gv_decitemp - tx_io .

ENDMETHOD.

METHOD reciprocal.

gv_decimal = 1 / (tx_io) .

ENDMETHOD.

METHOD anzeigen_temp.

CASE gv_auswahl.

WHEN 'PLUS'.

gv_optemp = '+'.

WHEN 'MINUS'.

gv_optemp = '-'.

WHEN 'DIV'.

gv_optemp = '/'.

```

    WHEN 'MAL'.
        gv_optemp = '*'.
    WHEN 'WURZEL'.
        gv_optemp = 'W'.
    WHEN OTHERS.
    ENDCASE.

IF gv_auswahl eq 'FREI'.
    tx_temp = tx_io.
else.
    tx_temp = tx_temp && gv_temp_zahl1 && gv_optemp && tx_io.
ENDIF.
*   tx_temp = tx_temp && gv_temp_zahl1 && gv_optemp && tx_io.

ENDMETHOD.

METHOD c_ce.
    CLEAR: tx_io, gv_decimal, gv_ergebnis, tx_temp, gv_tempd, gv_tempe, gv_temp_zahl1,
gv_counter.

ENDMETHOD.

METHOD reset.
    IF gv_flag_nummer NE 'X' .
        me->c_ce( ).
    ENDIF.
    gv_flag_nummer = 'X'.
ENDMETHOD.

```

METHOD eingabe.

```
DATA: "tx_io      TYPE char40,
lv_str1          TYPE string,
lv_str2          TYPE string,
lv_zahl1         TYPE string,
lv_zahl2         TYPE string,
lv_lenth         TYPE i,
lv_indexof       TYPE i,
lv_operator      TYPE c1,
lv_c1            TYPE i,
lv_c2            TYPE i,
lv_c3            TYPE i,
lv_c4            TYPE i,
lv_1cm           TYPE i,
lv_2cd           TYPE i,
lv_3cp           TYPE i,
lv_4cm           TYPE i,
lv_count_all     TYPE i,
lv_last          TYPE char20,
lv_erste         TYPE char20,
lv_index         TYPE i,
lv_c1t           TYPE i,
lv_c2t           TYPE i,
lv_c3t           TYPE i,
lv_c4t           TYPE i,
lv_nicht_erlaubt TYPE i,
```

```
lv_nicht_erlaubt_offset_mal TYPE i,  
lv_nicht_erlaubt_offset_div TYPE i,  
lv_nicht_erlaubt_offset_plus TYPE i,  
lv_nicht_erlaubt_offset_minus TYPE i.
```

```
IF strlen( tx_io ) EQ 0.
```

```
MESSAGE: 'Erstmal geben Aufgabe danach Eingabe ausdrücken' TYPE 'I'.
```

```
LEAVE PROGRAM.
```

```
ENDIF.
```

```
FIND REGEX '[a-zA-Z:.,;#~'!"$$%&(){}><|}]' IN tx_io MATCH COUNT lv_nicht_erlaubt.
```

```
IF sy-subrc EQ 0.
```

```
MESSAGE: 'Buchstaben oder Sonderzeichen sind nicht erlaubt!' TYPE 'E'.
```

```
*WAIT UP TO 10 SECONDS.
```

```
LEAVE PROGRAM.
```

```
ENDIF.
```

```
FIND REGEX '\+{2}' IN tx_io MATCH COUNT lv_nicht_erlaubt.
```

```
IF sy-subrc EQ 0.
```

```
MESSAGE: 'Nebeneinander Operatoren sind nicht erlaubt!' TYPE 'E'.
```

```
ENDIF.
```

```
FIND REGEX '\-{2}' IN tx_io MATCH COUNT lv_nicht_erlaubt.
```

```
IF sy-subrc EQ 0.
```

```
MESSAGE: 'Nebeneinander Operatoren sind nicht erlaubt!' TYPE 'E'.
```

```
ENDIF.
```

FIND REGEX '*{2}' IN tx_io MATCH COUNT lv_nicht_erlaubt.

IF sy-subrc EQ 0.

MESSAGE: 'Nebeneinander Operatoren sind nicht erlaubt!' TYPE 'E'.

ENDIF.

FIND REGEX '\{2}' IN tx_io MATCH COUNT lv_nicht_erlaubt.

IF sy-subrc EQ 0.

MESSAGE: 'Nebeneinander Operatoren sind nicht erlaubt!' TYPE 'E'.

ENDIF.

lv_erste = tx_io+0(1).

IF lv_erste EQ '/' OR lv_erste EQ '*' OR lv_erste EQ '+' OR lv_erste EQ '-'.

MESSAGE:'Operatoren sind auf letzte Position nicht erlaubt.' TYPE 'E'.

ENDIF.

lv_lenth = strlen(tx_io) - 1.

lv_last = tx_io+lv_lenth(1).

IF lv_last EQ '/' OR lv_last EQ '*' OR lv_last EQ '+' OR lv_last EQ '-'.

MESSAGE:'Operatoren sind auf letzte Position nicht erlaubt.' TYPE 'E'.

ENDIF.

CONDENSE tx_io NO-GAPS.


```

DATA: BEGIN OF ms_zahl,
      lt_zahl TYPE char20,
      END OF ms_zahl,
      mt_zahl LIKE TABLE OF ms_zahl.

```

```

FIND ALL OCCURRENCES OF '*' IN tx_io MATCH COUNT lv_c1.
FIND ALL OCCURRENCES OF '/' IN tx_io MATCH COUNT lv_c2.
FIND ALL OCCURRENCES OF '+' IN tx_io MATCH COUNT lv_c3.
FIND ALL OCCURRENCES OF '-' IN tx_io MATCH COUNT lv_c4.

lv_c1t = lv_c1.
lv_c2t = lv_c2.
lv_c3t = lv_c3.
lv_c4t = lv_c4.

```

```

lv_count_all = lv_c1 + lv_c2 + lv_c3 + lv_c4 .

```

* Split Formular

```

DO lv_count_all TIMES.
  IF lv_c1 GT 0.
    FIND FIRST OCCURRENCE OF '*' IN tx_io MATCH OFFSET lv_1cm.
  ENDIF.
  IF lv_c2 GT 0.
    FIND FIRST OCCURRENCE OF '/' IN tx_io MATCH OFFSET lv_2cd.
  ENDIF.
  IF lv_c3 GT 0.
    FIND FIRST OCCURRENCE OF '+' IN tx_io MATCH OFFSET lv_3cp.

```

ENDIF.

IF lv_c4 GT 0.

FIND FIRST OCCURRENCE OF '-' IN tx_io MATCH OFFSET lv_4cm.

ENDIF.

IF lv_1cm EQ 0.

lv_1cm = strlen(tx_io).

ENDIF.

IF lv_2cd EQ 0.

lv_2cd = strlen(tx_io).

ENDIF.

IF lv_3cp EQ 0.

lv_3cp = strlen(tx_io).

ENDIF.

IF lv_4cm EQ 0.

lv_4cm = strlen(tx_io).

ENDIF.

IF lv_1cm GT 0 AND lv_1cm LT lv_2cd AND lv_1cm LT lv_3cp AND lv_1cm LT lv_4cm.

SPLIT tx_io AT '*' INTO lv_str1 lv_str2.

APPEND lv_str1 TO mt_zahl.

APPEND '*' TO mt_zahl.

tx_io = lv_str2.

CLEAR lv_str1.

lv_c1 = lv_c1 - 1 .

lv_1cm = strlen(tx_io).

ELSEIF lv_2cd GT 0 AND lv_2cd LT lv_1cm AND lv_2cd LT lv_3cp AND lv_2cd LT lv_4cm.

```

SPLIT tx_io AT '/' INTO lv_str1 lv_str2.
APPEND lv_str1 TO mt_zahl.
APPEND '/' TO mt_zahl.
tx_io = lv_str2 .
CLEAR lv_str1.
lv_c2 = lv_c2 - 1 .
lv_2cd = strlen( tx_io ).
ELSEIF lv_3cp GT 0 AND lv_3cp LT lv_1cm AND lv_3cp LT lv_2cd AND lv_3cp LT
lv_4cm.

```

```

SPLIT tx_io AT '+' INTO lv_str1 lv_str2.
APPEND lv_str1 TO mt_zahl.
APPEND '+' TO mt_zahl.
tx_io = lv_str2.
CLEAR lv_str1.
lv_c3 = lv_c3 - 1 .
lv_3cp = strlen( tx_io ).
ELSEIF lv_4cm GT 0 AND lv_4cm LT lv_1cm AND lv_4cm LT lv_2cd AND lv_4cm LT
lv_3cp.

```

```

SPLIT tx_io AT '-' INTO lv_str1 lv_str2.
APPEND lv_str1 TO mt_zahl.
APPEND '-' TO mt_zahl.

```

```

tx_io = lv_str2 .
CLEAR lv_str1.
lv_c4 = lv_c4 - 1 .

```

lv_4cm = strlen(tx_io).

ENDIF.

lv_last = lv_c1 + lv_c2 + lv_c3 + lv_c4.

IF lv_last EQ 0.

CONDENSE lv_str2 NO-GAPS.

APPEND lv_str2 TO mt_zahl.

ENDIF.

ENDDO.

*Mal aufgaben

DO lv_c1t TIMES.

READ TABLE mt_zahl INTO ms_zahl WITH KEY lt_zahl = '*'.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix - 1).

lv_zahl1 = ms_zahl-lt_zahl.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix + 2).

lv_zahl2 = ms_zahl-lt_zahl.

IF lv_zahl1 EQ '*' OR lv_zahl1 EQ '/' OR lv_zahl1 EQ '+' OR lv_zahl1 EQ '-' OR

lv_zahl2 EQ '*' OR lv_zahl2 EQ '/' OR lv_zahl2 EQ '+' OR lv_zahl2 EQ '-'.

MESSAGE: 'Eingegebenes Formula ist ungültig' TYPE 'E'.

ENDIF.

DELETE mt_zahl INDEX sy-tabix.

DELETE mt_zahl INDEX sy-tabix - 1 .

DELETE mt_zahl INDEX sy-tabix - 2 .

lv_index = sy-tabix - 2 .

*write: / 'Zahl', lv_zahl1 , lv_zahl2 , / .

*delete mt_zahl INDEX sy-tabix + 1.

gv_decimal = lv_zahl1 * lv_zahl2.

INSERT gv_decimal INTO mt_zahl INDEX lv_index.

ENDDO.

*Div Aufgaben

DO lv_c2t TIMES.

READ TABLE mt_zahl INTO ms_zahl WITH KEY lt_zahl = '/'.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix - 1).

lv_zahl1 = ms_zahl-lt_zahl.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix + 2).

lv_zahl2 = ms_zahl-lt_zahl.

IF lv_zahl1 EQ '*' OR lv_zahl1 EQ '/' OR lv_zahl1 EQ '+' OR lv_zahl1 EQ '-' OR

lv_zahl2 EQ '*' OR lv_zahl2 EQ '/' OR lv_zahl2 EQ '+' OR lv_zahl2 EQ '-'.

MESSAGE: 'Eingegebenes Formula ist ungültig' TYPE 'E'.

ENDIF.

DELETE mt_zahl INDEX sy-tabix.

DELETE mt_zahl INDEX sy-tabix - 1 .

DELETE mt_zahl INDEX sy-tabix - 2 .

lv_index = sy-tabix - 2 .

*delete mt_zahl INDEX sy-tabix + 1.

IF lv_zahl2 EQ 0.

MESSAGE: 'Geteilt 0 ist unmöglich' TYPE 'E'.

LEAVE PROGRAM.

ENDIF.

gv_decimal = lv_zahl1 / lv_zahl2.

INSERT gv_decimal INTO mt_zahl INDEX lv_index.

ENDDO.

*add Aufgaben

DO lv_c3t TIMES.

READ TABLE mt_zahl INTO ms_zahl WITH KEY lt_zahl = '+'.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix - 1).

lv_zahl1 = ms_zahl-lt_zahl.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix + 2).

lv_zahl2 = ms_zahl-lt_zahl.

IF lv_zahl1 EQ '*' OR lv_zahl1 EQ '/' OR lv_zahl1 EQ '+' OR lv_zahl1 EQ '-' OR
lv_zahl2 EQ '*' OR lv_zahl2 EQ '/' OR lv_zahl2 EQ '+' OR lv_zahl2 EQ '-'.

MESSAGE: 'Eingegebenes Formula ist ungültig' TYPE 'E'.

ENDIF.

DELETE mt_zahl INDEX sy-tabix.

DELETE mt_zahl INDEX sy-tabix - 1 .

DELETE mt_zahl INDEX sy-tabix - 2 .

lv_index = sy-tabix - 2 .

gv_decimal = lv_zahl1 + lv_zahl2.

INSERT gv_decimal INTO mt_zahl INDEX lv_index.

ENDDO.

*Subt Aufgaben

DO lv_c4t TIMES.

READ TABLE mt_zahl INTO ms_zahl WITH KEY lt_zahl = '- '.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix - 1).

lv_zahl1 = ms_zahl-lt_zahl.

READ TABLE mt_zahl INTO ms_zahl INDEX (sy-tabix + 2).

lv_zahl2 = ms_zahl-lt_zahl.

IF lv_zahl1 EQ '*' OR lv_zahl1 EQ '/' OR lv_zahl1 EQ '+' OR lv_zahl1 EQ '-' OR
lv_zahl2 EQ '*' OR lv_zahl2 EQ '/' OR lv_zahl2 EQ '+' OR lv_zahl2 EQ '-'.

MESSAGE: 'Eingegebenes Formula ist ungültig' TYPE 'E'.

ENDIF.

DELETE mt_zahl INDEX sy-tabix.

DELETE mt_zahl INDEX sy-tabix - 1 .

DELETE mt_zahl INDEX sy-tabix - 2 .

lv_index = sy-tabix - 2 .

gv_decimal = lv_zahl1 - lv_zahl2.

INSERT gv_decimal INTO mt_zahl INDEX lv_index.

ENDDO.

LOOP AT mt_zahl INTO ms_zahl.

gv_decimal = ms_zahl-lt_zahl.

gv_ergebnis = ms_zahl-lt_zahl.

ENDLOOP.

ENDMETHOD.

ENDCLASS.

&-----

*& Include /LWVH/TASCHENRECHNERDYNPRO_I01

&-----

MODULE user_command_0111 INPUT.

DATA: lr_object TYPE REF TO lcl_taschenrechner.

CREATE OBJECT lr_object.

CASE sy-ucomm.

WHEN enter.

 lr_object->anzeigen_temp().

 lr_object->eingabe().

 IF gv_decimal MOD 1 GT 0.

 tx_io = gv_decimal .

 ELSE.

 tx_io = gv_ergebnis.

 ENDIF.

WHEN OTHERS.

ENDCASE.

CASE ok_code.

WHEN 'BACK' OR 'CANCEL' OR 'EXIT'.

 LEAVE PROGRAM.

WHEN 'ZERO'.


```
lr_object->reset( ).  
tx_io = tx_io && '0' .
```

WHEN 'EINS'.

```
lr_object->reset( ).  
tx_io = tx_io && '1' .
```

WHEN 'ZWEI'.

```
lr_object->reset( ).  
tx_io = tx_io && '2' .
```

WHEN 'DREI'.

```
lr_object->reset( ).  
tx_io = tx_io && '3' .
```

WHEN 'VIER'.

```
lr_object->reset( ).  
tx_io = tx_io && '4' .
```

WHEN 'FUNF'.

```
lr_object->reset( ).  
tx_io = tx_io && '5' .
```

WHEN 'SECHS'.

```
lr_object->reset( ).  
tx_io = tx_io && '6' .
```

WHEN 'SIEBEN'.

tx_io = tx_io && '7' .

WHEN 'ACHT'.

lr_object->reset().

tx_io = tx_io && '8' .

lr_object->reset().

WHEN 'NEUN'.

lr_object->reset().

tx_io = tx_io && '9' .

WHEN 'KOMMA'.

tx_io = tx_io && '.' .

WHEN 'RETURN'.

IF strlen(tx_io) GT 0.

tx_io = substring(val = tx_io off = 0 len = strlen(tx_io) - 1).

ENDIF.

IF strlen(tx_temp) GT 0.

tx_temp = substring(val = tx_temp off = 0 len = strlen(tx_temp) - 1).

ENDIF.

* SHIFT tx_io RIGHT by 1 PLACES CIRCULAR.

* tx_io = substring(val = tx_io off = strlen(tx_io) - 1 len = 1).

WHEN 'GLEICH' OR 'X'. " or 'ENTER' or enter.

gv_counter = gv_counter + 1.

lr_object->haupt().

IF gv_decimal MOD 1 GT 0.

tx_io = gv_decimal .

ELSE.

```

    tx_io = gv_ergebnis.

ENDIF.

CLEAR: tx_temp, gv_counter .

gv_flag_nummer = ' '.

WHEN 'PLUS' OR 'MINUS' OR 'MAL' OR 'DIV'. "OR 'WURZEL'.
*   TRY.

    gv_auswahl = ok_code .
    gv_counter = gv_counter + 1 .

    gv_temp_zahl1 = tx_io .
    lr_object->haupt( ).

    lr_object->anzeigen_temp( ).
    gv_flag_nummer = 'X'.
WHEN 'WURZEL'.

    gv_auswahl = ok_code.
    gv_ergebnis = tx_io .
    gv_decimal = tx_io .

    gv_temp_zahl1 = tx_io .
    CLEAR tx_io.
    lr_object->anzeigen_temp( ).
    gv_flag_nummer = 'X'.
WHEN 'PROZENT'.

    gv_auswahl = ok_code.

```

```

gv_decimal = tx_io .
gv_temp_zahl1 = tx_io .
CLEAR tx_io.
lr_object->anzeigen_temp( ).
gv_flag_nummer = 'X'.
WHEN 'EINGABE'.
*   tx_io = tx_temp.
*   gv_temp_zahl1 = tx_io .
*   lr_object->anzeigen_temp( ).

lr_object->eingabe( ).
IF gv_decimal MOD 1 GT 0.
    tx_io = gv_decimal .
ELSE.
    tx_io = gv_ergebnis.
ENDIF.
WHEN 'PLMN'.
    tx_io = tx_io * -1 .
    gv_flag_nummer = 'X'.
WHEN 'MS'.
    gv_auswahl = ok_code.
    gv_ergebtemp = tx_io.
    gv_decitemp = tx_io.
    CLEAR tx_io.
    gv_flag_nummer = 'X'.
WHEN 'MR'.
    IF gv_decitemp MOD 1 GT 0.
        tx_io = gv_decitemp .

```

```

ELSE.

    tx_io = gv_ergebtemp .

ENDIF.

gv_flag_nummer = 'X'.

WHEN 'C' OR 'CE'.

*   lr_object->c_ce( ).

    CLEAR: tx_io, gv_decimal, gv_ergebnis, tx_temp, gv_tempd, gv_tempe, gv_temp_zahl1,
gv_counter.

    gv_flag_nummer = 'X'.

WHEN 'MC'.

    CLEAR : tx_io, gv_decitemp , gv_ergebtemp .

WHEN 'FREI'.

    gv_auswahl = ok_code.

WHEN OTHERS.

ENDCASE.

ENDMODULE.

*&-----*
*&   Module CHECK_FIELD INPUT
*&-----*
*   text
*-----*

MODULE check_field INPUT.

    if tx_io ca sy-abcde.

        MESSAGE 'Bitte keine Buchstaben eingeben' type 'E'.

    endif.

ENDMODULE.

```

Erstellen eines kurzen Konzepts zur folgenden Problemstellung: (Kalender)

Einführung

Fachbereich 102.0 möchte einen „Programm“. Es soll ein Kalender. Es soll aber eine freie Eingabe sein, welche Jahr eingegeben werden kann. Es soll deutsche Ferientage z.B. Ostern, Einheit, Weichmachten usw. anzeigen und highlighten.

Ist-Stand

Zurzeit gibt es keine Dasselbe oder Ähnliches.

Projektziel

Dieses Konzept bietet ein Konzept, das angezeigt wird, wie die Rechnung Aufgabe ist.

Beschreibung des Geschäftsprozess

Das Programm hilft dabei, dass ABAP-Kenntnisse verbessert werden kann.

Programm /LWVH/KALENDEROOP



Willkommen Kalender Programm.

Jahr

2024

Programm /LWVH/KALENDEROOP

Programm /LWVH/KALENDEROOP

01.01.2024	01.02.2024	01.03.2024	01.04.2024	01.05.2024	01.06.2024	01.07.2024	01.08.2024	01.09.2024	01.10.2024	01.11.2024	01.12.2024
02.01.2024	02.02.2024	02.03.2024	02.04.2024	02.03.2024	02.06.2024	02.07.2024	02.08.2024	02.09.2024	02.10.2024	02.11.2024	02.12.2024
03.01.2024	03.02.2024	03.03.2024	03.04.2024	03.03.2024	03.06.2024	03.07.2024	03.08.2024	03.09.2024	03.10.2024	03.11.2024	03.12.2024
04.01.2024	04.02.2024	04.03.2024	04.04.2024	04.03.2024	04.06.2024	04.07.2024	04.08.2024	04.09.2024	04.10.2024	04.11.2024	04.12.2024
05.01.2024	05.02.2024	05.03.2024	05.04.2024	05.03.2024	05.06.2024	05.07.2024	05.08.2024	05.09.2024	05.10.2024	05.11.2024	05.12.2024
06.01.2024	06.02.2024	06.03.2024	06.04.2024	06.03.2024	06.06.2024	06.07.2024	06.08.2024	06.09.2024	06.10.2024	06.11.2024	06.12.2024
07.01.2024	07.02.2024	07.03.2024	07.04.2024	07.03.2024	07.06.2024	07.07.2024	07.08.2024	07.09.2024	07.10.2024	07.11.2024	07.12.2024
08.01.2024	08.02.2024	08.03.2024	08.04.2024	08.03.2024	08.06.2024	08.07.2024	08.08.2024	08.09.2024	08.10.2024	08.11.2024	08.12.2024
09.01.2024	09.02.2024	09.03.2024	09.04.2024	09.03.2024	09.06.2024	09.07.2024	09.08.2024	09.09.2024	09.10.2024	09.11.2024	09.12.2024
10.01.2024	10.02.2024	10.03.2024	10.04.2024	10.03.2024	10.06.2024	10.07.2024	10.08.2024	10.09.2024	10.10.2024	10.11.2024	10.12.2024
11.01.2024	11.02.2024	11.03.2024	11.04.2024	11.03.2024	11.06.2024	11.07.2024	11.08.2024	11.09.2024	11.10.2024	11.11.2024	11.12.2024
12.01.2024	12.02.2024	12.03.2024	12.04.2024	12.03.2024	12.06.2024	12.07.2024	12.08.2024	12.09.2024	12.10.2024	12.11.2024	12.12.2024
13.01.2024	13.02.2024	13.03.2024	13.04.2024	13.03.2024	13.06.2024	13.07.2024	13.08.2024	13.09.2024	13.10.2024	13.11.2024	13.12.2024
14.01.2024	14.02.2024	14.03.2024	14.04.2024	14.03.2024	14.06.2024	14.07.2024	14.08.2024	14.09.2024	14.10.2024	14.11.2024	14.12.2024
15.01.2024	15.02.2024	15.03.2024	15.04.2024	15.03.2024	15.06.2024	15.07.2024	15.08.2024	15.09.2024	15.10.2024	15.11.2024	15.12.2024
16.01.2024	16.02.2024	16.03.2024	16.04.2024	16.03.2024	16.06.2024	16.07.2024	16.08.2024	16.09.2024	16.10.2024	16.11.2024	16.12.2024
17.01.2024	17.02.2024	17.03.2024	17.04.2024	17.03.2024	17.06.2024	17.07.2024	17.08.2024	17.09.2024	17.10.2024	17.11.2024	17.12.2024
18.01.2024	18.02.2024	18.03.2024	18.04.2024	18.03.2024	18.06.2024	18.07.2024	18.08.2024	18.09.2024	18.10.2024	18.11.2024	18.12.2024
19.01.2024	19.02.2024	19.03.2024	19.04.2024	19.03.2024	19.06.2024	19.07.2024	19.08.2024	19.09.2024	19.10.2024	19.11.2024	19.12.2024
20.01.2024	20.02.2024	20.03.2024	20.04.2024	20.03.2024	20.06.2024	20.07.2024	20.08.2024	20.09.2024	20.10.2024	20.11.2024	20.12.2024
21.01.2024	21.02.2024	21.03.2024	21.04.2024	21.03.2024	21.06.2024	21.07.2024	21.08.2024	21.09.2024	21.10.2024	21.11.2024	21.12.2024
22.01.2024	22.02.2024	22.03.2024	22.04.2024	22.03.2024	22.06.2024	22.07.2024	22.08.2024	22.09.2024	22.10.2024	22.11.2024	22.12.2024
23.01.2024	23.02.2024	23.03.2024	23.04.2024	23.03.2024	23.06.2024	23.07.2024	23.08.2024	23.09.2024	23.10.2024	23.11.2024	23.12.2024
24.01.2024	24.02.2024	24.03.2024	24.04.2024	24.03.2024	24.06.2024	24.07.2024	24.08.2024	24.09.2024	24.10.2024	24.11.2024	24.12.2024
25.01.2024	25.02.2024	25.03.2024	25.04.2024	25.03.2024	25.06.2024	25.07.2024	25.08.2024	25.09.2024	25.10.2024	25.11.2024	25.12.2024
26.01.2024	26.02.2024	26.03.2024	26.04.2024	26.03.2024	26.06.2024	26.07.2024	26.08.2024	26.09.2024	26.10.2024	26.11.2024	26.12.2024
27.01.2024	27.02.2024	27.03.2024	27.04.2024	27.03.2024	27.06.2024	27.07.2024	27.08.2024	27.09.2024	27.10.2024	27.11.2024	27.12.2024
28.01.2024	28.02.2024	28.03.2024	28.04.2024	28.03.2024	28.06.2024	28.07.2024	28.08.2024	28.09.2024	28.10.2024	28.11.2024	28.12.2024
29.01.2024		29.03.2024	29.04.2024	29.03.2024	29.06.2024	29.07.2024	29.08.2024	29.09.2024	29.10.2024	29.11.2024	29.12.2024
30.01.2024		30.03.2024	30.04.2024	30.03.2024	30.06.2024	30.07.2024	30.08.2024	30.09.2024	30.10.2024	30.11.2024	30.12.2024
31.01.2024		31.03.2024		31.03.2024		31.07.2024	31.08.2024		31.10.2024		31.12.2024

Programm /LWVH/KALENDEROOP



Willkommen Kalender Programm.

Jahr

2001

Programm /LWVH/KALENDEROOP

Programm /LWVH/KALENDEROOP

01.01.2001	01.02.2001	01.03.2001	01.04.2001	01.05.2001	01.06.2001	01.07.2001	01.08.2001	01.09.2001	01.10.2001	01.11.2001	01.12.2001
02.01.2001	02.02.2001	02.03.2001	02.04.2001	02.03.2001	02.06.2001	02.07.2001	02.08.2001	02.09.2001	02.10.2001	02.11.2001	02.12.2001
03.01.2001	03.02.2001	03.03.2001	03.04.2001	03.03.2001	03.06.2001	03.07.2001	03.08.2001	03.09.2001	03.10.2001	03.11.2001	03.12.2001
04.01.2001	04.02.2001	04.03.2001	04.04.2001	04.03.2001	04.06.2001	04.07.2001	04.08.2001	04.09.2001	04.10.2001	04.11.2001	04.12.2001
05.01.2001	05.02.2001	05.03.2001	05.04.2001	05.03.2001	05.06.2001	05.07.2001	05.08.2001	05.09.2001	05.10.2001	05.11.2001	05.12.2001
06.01.2001	06.02.2001	06.03.2001	06.04.2001	06.03.2001	06.06.2001	06.07.2001	06.08.2001	06.09.2001	06.10.2001	06.11.2001	06.12.2001
07.01.2001	07.02.2001	07.03.2001	07.04.2001	07.03.2001	07.06.2001	07.07.2001	07.08.2001	07.09.2001	07.10.2001	07.11.2001	07.12.2001
08.01.2001	08.02.2001	08.03.2001	08.04.2001	08.03.2001	08.06.2001	08.07.2001	08.08.2001	08.09.2001	08.10.2001	08.11.2001	08.12.2001
09.01.2001	09.02.2001	09.03.2001	09.04.2001	09.03.2001	09.06.2001	09.07.2001	09.08.2001	09.09.2001	09.10.2001	09.11.2001	09.12.2001
10.01.2001	10.02.2001	10.03.2001	10.04.2001	10.03.2001	10.06.2001	10.07.2001	10.08.2001	10.09.2001	10.10.2001	10.11.2001	10.12.2001
11.01.2001	11.02.2001	11.03.2001	11.04.2001	11.03.2001	11.06.2001	11.07.2001	11.08.2001	11.09.2001	11.10.2001	11.11.2001	11.12.2001
12.01.2001	12.02.2001	12.03.2001	12.04.2001	12.03.2001	12.06.2001	12.07.2001	12.08.2001	12.09.2001	12.10.2001	12.11.2001	12.12.2001
13.01.2001	13.02.2001	13.03.2001	13.04.2001	13.03.2001	13.06.2001	13.07.2001	13.08.2001	13.09.2001	13.10.2001	13.11.2001	13.12.2001
14.01.2001	14.02.2001	14.03.2001	14.04.2001	14.03.2001	14.06.2001	14.07.2001	14.08.2001	14.09.2001	14.10.2001	14.11.2001	14.12.2001
15.01.2001	15.02.2001	15.03.2001	15.04.2001	15.03.2001	15.06.2001	15.07.2001	15.08.2001	15.09.2001	15.10.2001	15.11.2001	15.12.2001
16.01.2001	16.02.2001	16.03.2001	16.04.2001	16.03.2001	16.06.2001	16.07.2001	16.08.2001	16.09.2001	16.10.2001	16.11.2001	16.12.2001
17.01.2001	17.02.2001	17.03.2001	17.04.2001	17.03.2001	17.06.2001	17.07.2001	17.08.2001	17.09.2001	17.10.2001	17.11.2001	17.12.2001
18.01.2001	18.02.2001	18.03.2001	18.04.2001	18.03.2001	18.06.2001	18.07.2001	18.08.2001	18.09.2001	18.10.2001	18.11.2001	18.12.2001
19.01.2001	19.02.2001	19.03.2001	19.04.2001	19.03.2001	19.06.2001	19.07.2001	19.08.2001	19.09.2001	19.10.2001	19.11.2001	19.12.2001
20.01.2001	20.02.2001	20.03.2001	20.04.2001	20.03.2001	20.06.2001	20.07.2001	20.08.2001	20.09.2001	20.10.2001	20.11.2001	20.12.2001
21.01.2001	21.02.2001	21.03.2001	21.04.2001	21.03.2001	21.06.2001	21.07.2001	21.08.2001	21.09.2001	21.10.2001	21.11.2001	21.12.2001
22.01.2001	22.02.2001	22.03.2001	22.04.2001	22.03.2001	22.06.2001	22.07.2001	22.08.2001	22.09.2001	22.10.2001	22.11.2001	22.12.2001
23.01.2001	23.02.2001	23.03.2001	23.04.2001	23.03.2001	23.06.2001	23.07.2001	23.08.2001	23.09.2001	23.10.2001	23.11.2001	23.12.2001
24.01.2001	24.02.2001	24.03.2001	24.04.2001	24.03.2001	24.06.2001	24.07.2001	24.08.2001	24.09.2001	24.10.2001	24.11.2001	24.12.2001
25.01.2001	25.02.2001	25.03.2001	25.04.2001	25.03.2001	25.06.2001	25.07.2001	25.08.2001	25.09.2001	25.10.2001	25.11.2001	25.12.2001
26.01.2001	26.02.2001	26.03.2001	26.04.2001	26.03.2001	26.06.2001	26.07.2001	26.08.2001	26.09.2001	26.10.2001	26.11.2001	26.12.2001
27.01.2001	27.02.2001	27.03.2001	27.04.2001	27.03.2001	27.06.2001	27.07.2001	27.08.2001	27.09.2001	27.10.2001	27.11.2001	27.12.2001
28.01.2001	28.02.2001	28.03.2001	28.04.2001	28.03.2001	28.06.2001	28.07.2001	28.08.2001	28.09.2001	28.10.2001	28.11.2001	28.12.2001
29.01.2001		29.03.2001	29.04.2001	29.03.2001	29.06.2001	29.07.2001	29.08.2001	29.09.2001	29.10.2001	29.11.2001	29.12.2001
30.01.2001		30.03.2001	30.04.2001	30.03.2001	30.06.2001	30.07.2001	30.08.2001	30.09.2001	30.10.2001	30.11.2001	30.12.2001
31.01.2001		31.03.2001		31.03.2001		31.07.2001	31.08.2001		31.10.2001		31.12.2001

&-----

*& Include /LWVH/KALENDEROOP_TOP

&-----

CLASS lcl_kalenderoop DEFINITION.

PUBLIC SECTION.

METHODS: writemethode IMPORTING

```
i_jahr  TYPE i
i_osto1 TYPE d
i_osto2 TYPE d
i_ostkt TYPE d
i_monat TYPE i
i_nejhr TYPE d
i_arbt  TYPE d
i_einht TYPE d
i_weihn TYPE d
i_weihn2 TYPE d
i_weihn3 TYPE d
i_silve TYPE d
i_jan   TYPE d
i_feb   TYPE d
i_mar   TYPE d
i_mai   TYPE d
i_juni  TYPE d
i_juli  TYPE d
i_agst  TYPE d
i_sep   TYPE d
i_okt   TYPE d
i_nvbr  TYPE d
```

i_dzbr TYPE d

i_f28 TYPE d

i_a30 TYPE d

i_j30 TYPE d

i_s30 TYPE d

i_n30 TYPE d.

DATA: gv_neujr TYPE d VALUE '00010101',

gv_oster TYPE d VALUE '00010321',

gv_rosen TYPE d VALUE '00010201',

gv_arbt TYPE d VALUE '00010501',

gv_himme TYPE d VALUE '00010501',

gv_pfung TYPE d VALUE '00010501',

gv_einht TYPE d VALUE '00011003',

gv_weihn TYPE d VALUE '00011224',

gv_silve TYPE d VALUE '00011231',

gv_jan TYPE d VALUE '00010101',

gv_feb TYPE d VALUE '00010201',

gv_maerz TYPE d VALUE '00010301',

gv_april TYPE d VALUE '00010401',

gv_mai TYPE d VALUE '00010501',

gv_juni TYPE d VALUE '00010601',

gv_juli TYPE d VALUE '00010701',

gv_august TYPE d VALUE '00010801',

gv_septembr TYPE d VALUE '00010901',

```
gv_oktober TYPE d VALUE '00011001',  
gv_novmbr  TYPE d VALUE '00011101',  
gv_dezember TYPE d VALUE '00011201'.
```

DATA: gv_monat TYPE i.

METHODS: osternrechnen IMPORTING

```
    i_jahr TYPE i  
EXPORTING  
    e_tag_k TYPE d  "Karfreitag  
    e_tag_o1 TYPE d  "Ostern Sonntag  
    e_tag_o2 TYPE d. "Ostern Montag
```

METHODS: monatkalkulation IMPORTING

```
    i_jahr TYPE i  
EXPORTING  
    e_monat TYPE i.
```

METHODS: constantfeierntagekalkulation IMPORTING

```
    i_jahr TYPE i  
EXPORTING  
    e_neujhr TYPE d  
    e_arbt  TYPE d  
    e_einht TYPE d  
    e_weihn TYPE d  
    e_weihn2 TYPE d  
    e_weihn3 TYPE d  
    e_silve TYPE d
```

e_jan TYPE d
e_feb TYPE d
e_maer TYPE d
e_mai TYPE d
e_aprl TYPE d
e_juni TYPE d
e_juli TYPE d
e_agst TYPE d
e_sep TYPE d
e_oktr TYPE d
e_nvbr TYPE d
e_dzmr TYPE d
e_f28 TYPE d
e_a30 TYPE d
e_j30 TYPE d
e_s30 TYPE d
e_n30 TYPE d.

METHODS: check_holiday IMPORTING i_datum TYPE d RETURNING
VALUE(is_holiday) TYPE abap_bool.

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

&-----

*& Include /LWVH/KALENDEROOP_SEL

&-----

SELECTION-SCREEN BEGIN OF BLOCK frame1 WITH FRAME TITLE TEXT-001.

PARAMETERS: p_jahr TYPE i OBLIGATORY.

SELECTION-SCREEN END OF BLOCK frame1.

&-----

*& Include /LWVH/KALENDEROOP_P01

&-----

CLASS lcl_kalenderoop IMPLEMENTATION.

METHOD check_holiday.

*data: datum TYPE d VALUE '20230101'.

*datum = i_datum.

* IF (datum(is_valid)) EQ abap_true.

* " Ist das ein Feiertag, wenn ja:

* is_holiday = abap_true.

*

* " ansonsten ->

```

*

*   is_holiday = abap_false.

*

*   ELSE.

*

*   ENDIF.

*

*

*data lt_statische_feiertage type TABLE OF d.

*

*   APPEND p_jahr && '0101' TO lt_statische_feiertage.

*   APPEND p_jahr && '1231' TO lt_statische_feiertage.

*

*

*   TRY.

*lv_feiertag = lt_statische_feiertage[ table_line = lv_datum ].

*

*   CATCH cx_sy_itab_line_not_found.

*       " kein statisch Feiertag.

*   ENDTRY.

*

*

*while lv_monat(is_valid) .

*

*lv_monat = 1 .

*

*while lv_datum(is_valid).

*

```

```

*lv_i = 1.
*
*lv_datum = '2023' && lv_monat && lv_i.
*
*if is_date_valid( lv_datum ) eq abap_true.
*
*
* if is_holiday(lv_datum) eq abap_true.
*write: lv_datum DD/MM/YYYY color 3.
*
* else.
*write: lv_datum DD/MM/YYYY.
*endif.
*else.
*exit.
*endif.
*
*
*
*lv_i = lv_i + 1.
*
*endwhile.
*
*lv_monat = lv_monat + 1.
*
*endwhile.

```

ENDMETHOD.

METHOD writemethode .

```
* IF me->check_holiday( sy-datum ) EQ abap_true.  
*   WRITE: ls_datum-januar DD/MM/YYYY COLOR 3.  
* ELSE.  
*   WRITE: ls_datum-januar DD/MM/YYYY .  
* ENDIF.
```

DO 31 TIMES.

APPEND ls_datum TO lt_datum.

IF ls_datum-januar NE lv_nejhr .

WRITE: ls_datum-januar DD/MM/YYYY .

ELSE.

WRITE: ls_datum-januar DD/MM/YYYY COLOR 3.

ENDIF.

IF ls_datum-februar <= i_f28.

WRITE: ls_datum-februar DD/MM/YYYY .

ELSE.

WRITE: ' ' .

ENDIF.

IF ls_datum-maerz NE lv_ostkt AND ls_datum-maerz NE lv_osto1 AND ls_datum-maerz
NE lv_osto2 .

WRITE: ls_datum-maerz DD/MM/YYYY .

ELSE. "ls_datum-maerz EQ lv_osto1 OR ls_datum-maerz EQ lv_osto2 OR ls_datum-
maerz EQ lv_ostkt.

WRITE: ls_datum-maerz DD/MM/YYYY COLOR 3.

ENDIF.

IF ls_datum-april <= i_a30 AND ls_datum-april NE lv_ostkt AND ls_datum-april NE
lv_osto1 AND ls_datum-april NE lv_osto2 .

WRITE: ls_datum-april DD/MM/YYYY .

ELSEIF ls_datum-april GT i_a30.

WRITE: ' ' .

ELSEIF ls_datum-april EQ i_osto1.

WRITE: ls_datum-april DD/MM/YYYY COLOR 3.

ELSE. " ls_datum-april EQ lv_osto1 OR ls_datum-april EQ lv_osto2 OR ls_datum-april
EQ lv_ostkt.

WRITE: ls_datum-april DD/MM/YYYY COLOR 3.

ENDIF.

IF ls_datum-mai NE lv_arbt .

WRITE: ls_datum-maerz DD/MM/YYYY .

ELSE.

WRITE: ls_datum-mai DD/MM/YYYY COLOR 3.

ENDIF.

IF ls_datum-juni <= i_j30.

WRITE: ls_datum-juni DD/MM/YYYY . .

ELSE.

WRITE: ' ' .

ENDIF.

WRITE: ls_datum-juli DD/MM/YYYY , ls_datum-august DD/MM/YYYY .

IF ls_datum-septembr <= i_s30.

WRITE: ls_datum-septembr DD/MM/YYYY . .

ELSE.

WRITE: ' ' .

ENDIF.

IF ls_datum-oktober NE lv_einht .

WRITE: ls_datum-oktober DD/MM/YYYY .

ELSE.

WRITE: ls_datum-oktober DD/MM/YYYY COLOR 3.

ENDIF.

IF ls_datum-novmbr <= i_n30.

WRITE: ls_datum-novmbr DD/MM/YYYY . .

ELSE.

WRITE: ' ' .

ENDIF.

```
IF ls_datum-dezember NE lv_weihn AND ls_datum-dezember NE lv_weihn2 AND  
ls_datum-dezember NE lv_weihn3 AND ls_datum-dezember NE lv_silve.
```

```
WRITE: ls_datum-dezember DD/MM/YYYY .
```

```
ELSE.
```

```
WRITE: ls_datum-dezember DD/MM/YYYY COLOR 3.
```

```
ENDIF.
```

```
CALL FUNCTION 'CALCULATE_DATE'
```

```
EXPORTING
```

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
```

```
months     = '0'      " Differenz in Monaten (positiv oder negativ)
```

```
start_date = ls_datum-januar " Basisdatum
```

```
IMPORTING
```

```
result_date = ls_datum-januar. " Ergebnisdatum
```

```
CALL FUNCTION 'CALCULATE_DATE'
```

```
EXPORTING
```

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
```

```
months     = '0'      " Differenz in Monaten (positiv oder negativ)
```

```
start_date = ls_datum-februar " Basisdatum
```

```
IMPORTING
```

```
result_date = ls_datum-februar. " Ergebnisdatum
```

```
CALL FUNCTION 'CALCULATE_DATE'
```

```
EXPORTING
```

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
```

```
months     = '0'      " Differenz in Monaten (positiv oder negativ)
```

start_date = ls_datum-maerz " Basisdatum

IMPORTING

result_date = ls_datum-maerz. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)

months = '0' " Differenz in Monaten (positiv oder negativ)

start_date = ls_datum-april " Basisdatum

IMPORTING

result_date = ls_datum-april. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)

months = '0' " Differenz in Monaten (positiv oder negativ)

start_date = ls_datum-mai " Basisdatum

IMPORTING

result_date = ls_datum-mai. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)

months = '0' " Differenz in Monaten (positiv oder negativ)

start_date = ls_datum-juni " Basisdatum

IMPORTING

result_date = ls_datum-juni. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)
months = '0' " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-juli " Basisdatum

IMPORTING

result_date = ls_datum-juli. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)
months = '0' " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-august " Basisdatum

IMPORTING

result_date = ls_datum-august. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '1' " Differenz in Tagen (positiv oder negativ)
months = '0' " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-septembr " Basisdatum

IMPORTING

result_date = ls_datum-septembr. " Ergebnisdatum

WRITE: / .

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
months     = '0'      " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-oktober  " Basisdatum
```

IMPORTING

```
result_date = ls_datum-oktober.  " Ergebnisdatum
```

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
months     = '0'      " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-novmbr  " Basisdatum
```

IMPORTING

```
result_date = ls_datum-novmbr.  " Ergebnisdatum
```

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

```
days      = '1'      " Differenz in Tagen (positiv oder negativ)
months     = '0'      " Differenz in Monaten (positiv oder negativ)
start_date = ls_datum-dezember  " Basisdatum
```

IMPORTING

```
result_date = ls_datum-dezember.  " Ergebnisdatum
```

*WRITE: / .

ENDDO.

ENDMETHOD .

*data: lv_monat TYPE i.

METHOD osternrechnen .

DATA: lv_a TYPE i,

lv_b TYPE i,

lv_c TYPE i,

lv_d TYPE i,

lv_dg TYPE i,

lv_e TYPE i,

lv_mg TYPE i,

lv_m TYPE i,

lv_ng TYPE i,

lv_s TYPE i,

lv_tagp TYPE i.

lv_a = p_jahr MOD 19 .

lv_b = p_jahr MOD 4 .

lv_c = p_jahr MOD 7 .

lv_m = (((8 * (p_jahr / 100)) + 13) / 25) - 2 .

lv_s = ((p_jahr / 100) - (p_jahr / 400)) - 2 .

lv_mg = (15 + lv_s - lv_m) MOD 30 .

lv_ng = (6 + lv_s) MOD 7 .

lv_d = (lv_mg + (19 * lv_a)) MOD 30 .

lv_dg = lv_d .

IF lv_dg EQ 29 .

lv_dg = 28 .

ELSEIF lv_dg EQ 28 AND lv_a >= 11 .

lv_dg = 27 .

ENDIF.

lv_e = (2 * lv_b + 4 * lv_c + 6 * lv_dg + lv_ng) MOD 7 .

lv_tagp = lv_dg + lv_e + 1 .

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = lv_tagp " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_oster " Basisdatum

IMPORTING

result_date = e_tag_o1. " Ergebnisdatum

* e_tag_o1 = gv_oster .

* e_tag_o1 = e_tag_o1 + lv_etag - 1 .

e_tag_o2 = e_tag_o1 + 1 .

e_tag_k = e_tag_o1 - 2 .

ENDMETHOD.

METHOD monatkalkulation .

gv_monat = p_jahr * 12 - 12 .

ENDMETHOD.

METHOD constantfeierntagekalkulation .

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_neujr " Basisdatum

IMPORTING

result_date = e_neujhr. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_arbt " Basisdatum

IMPORTING

result_date = e_arbt. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_einht " Basisdatum

IMPORTING

result_date = e_einht. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_weihn " Basisdatum

IMPORTING

result_date = e_weihn. " Ergebnisdatum

e_weihn2 = e_weihn + 1 .

e_weihn3 = e_weihn + 2 .

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_silve " Basisdatum

IMPORTING

result_date = e_silve. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_jan " Basisdatum

IMPORTING

result_date = e_jan. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_feb " Basisdatum

IMPORTING

result_date = e_feb. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_maerz " Basisdatum

IMPORTING

result_date = e_maer. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_april " Basisdatum

IMPORTING

result_date = e_aprl. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_mai " Basisdatum

IMPORTING

result_date = e_mai. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_juli " Basisdatum

IMPORTING

result_date = e_juli. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_juni " Basisdatum

IMPORTING

result_date = e_juni. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_august " Basisdatum

IMPORTING

result_date = e_agst. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_septembr " Basisdatum

IMPORTING

result_date = e_sep. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_oktober " Basisdatum

IMPORTING

result_date = e_oktr. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_novmbr " Basisdatum

IMPORTING

result_date = e_nvbr. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = '0' " Differenz in Tagen (positiv oder negativ)
months = gv_monat " Differenz in Monaten (positiv oder negativ)
start_date = gv_dezember " Basisdatum

IMPORTING

result_date = e_dzmr. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = 27 " Differenz in Tagen (positiv oder negativ)
months = gv_monat " Differenz in Monaten (positiv oder negativ)
start_date = gv_feb " Basisdatum

IMPORTING

result_date = e_f28. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = 29 " Differenz in Tagen (positiv oder negativ)
months = gv_monat " Differenz in Monaten (positiv oder negativ)
start_date = gv_april " Basisdatum

IMPORTING

result_date = e_a30. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = 29 " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_juni " Basisdatum

IMPORTING

result_date = e_j30. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = 29 " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_septembr " Basisdatum

IMPORTING

result_date = e_s30. " Ergebnisdatum

CALL FUNCTION 'CALCULATE_DATE'

EXPORTING

days = 29 " Differenz in Tagen (positiv oder negativ)

months = gv_monat " Differenz in Monaten (positiv oder negativ)

start_date = gv_novmbr " Basisdatum

IMPORTING

result_date = e_n30. " Ergebnisdatum

ENDMETHOD.

ENDCLASS.

&-----

*& Include /LWVH/KALENDEROOP_E01

&-----

DATA : lr_kalender TYPE REF TO lcl_kalenderoop.

CREATE OBJECT lr_kalender.

DATA: lv_osto1 TYPE d,

 lv_osto2 TYPE d,

 lv_ostkt TYPE d,

 lv_monat TYPE i,

 lv_nejhr TYPE d,

 lv_arbt TYPE d,

 lv_einht TYPE d,

 lv_weihn TYPE d,

 lv_weihn2 TYPE d,

 lv_weihn3 TYPE d,

lv_silve TYPE d,
lv_jan TYPE d,
lv_feb TYPE d,
lv_maer TYPE d,
lv_mai TYPE d,
lv_apri TYPE d,
lv_juni TYPE d,
lv_juli TYPE d,
lv_agst TYPE d,
lv_sep TYPE d,
lv_oktr TYPE d,
lv_nvbr TYPE d,
lv_dzmr TYPE d,
lv_f28 TYPE d,
lv_a30 TYPE d,
lv_j30 TYPE d,
lv_s30 TYPE d,
lv_n30 TYPE d.

lr_kalender->monatkalkulation(

EXPORTING

i_jahr = p_jahr

IMPORTING

e_monat = lv_monat).

lr_kalender->osternrechnen(

EXPORTING

i_jahr = p_jahr

IMPORTING

e_tag_o1 = lv_ostto1

e_tag_o2 = lv_ostto2

e_tag_k = lv_ostkt).

lv_kalender->constantfeierntagekalkulation(

EXPORTING

i_jahr = p_jahr

IMPORTING

e_neujhr = lv_nejhr

e_arbt = lv_arbt

e_einht = lv_einht

e_weihn = lv_weihn

e_weihn2 = lv_weihn2

e_weihn3 = lv_weihn3

e_silve = lv_silve

e_jan = lv_jan

e_feb = lv_feb

e_maer = lv_maer

e_mai = lv_mai

e_aprl = lv_aprl

e_juni = lv_juni

e_juli = lv_juli

e_agst = lv_agst

e_sep = lv_sep

e_oktr = lv_oktr

```
e_nvbr  = lv_nvbr
e_dzmr  = lv_dzmr
e_f28   = lv_f28
e_a30   = lv_a30
e_j30   = lv_j30
e_s30   = lv_s30
e_n30   = lv_n30
).
```

```
TYPES: BEGIN OF lsy_datum,
```

```
    januar TYPE d,
    februar TYPE d,
    maerz   TYPE d,
    april   TYPE d,
    mai     TYPE d,
    juni    TYPE d,
    juli    TYPE d,
    august  TYPE d,
    septembr TYPE d,
    oktober TYPE d,
    novmbr  TYPE d,
    dezember TYPE d,
END OF lsy_datum.
```

```
DATA: lv_dtm TYPE d,
      lt_datum TYPE TABLE OF lsy_datum,
      ls_datum LIKE LINE OF lt_datum.
```

ls_datum-januar = lv_jan .
ls_datum-februar = lv_feb .
ls_datum-maerz = lv_maer .
ls_datum-april = lv_aprl .
ls_datum-mai = lv_mai .
ls_datum-juni = lv_juni .
ls_datum-juli = lv_juli .
ls_datum-august = lv_agst .
ls_datum-septembr = lv_sep .
ls_datum-oktober = lv_oktr .
ls_datum-novmbr = lv_nvbr .
ls_datum-dezember = lv_dzmr .

lr_kalender->writemethode(

EXPORTING

i_jahr = p_jahr
i_osto1 = lv_osto1
i_osto2 = lv_osto2
i_ostkt = lv_ostkt
i_monat = lv_monat
i_nejhr = lv_nejhr
i_arbt = lv_arbt
i_einht = lv_einht
i_weihn = lv_weihn
i_weihn2 = lv_weihn2
i_weihn3 = lv_weihn3

```
i_silve = lv_silve  
i_jan   = ls_datum-januar  
i_feb   = ls_datum-februar  
i_mar   = lv_maer  
i_mai   = lv_mai  
i_juni  = lv_juni  
i_juli  = lv_juli  
i_agst  = lv_agst  
i_sep   = lv_sep  
i_okt   = lv_oktr  
i_nvbr  = lv_nvbr  
i_dzbr  = lv_dzmr  
i_f28   = lv_f28  
i_a30   = lv_a30  
i_j30   = lv_j30  
i_s30   = lv_s30  
i_n30   = lv_n30  
).
```

AT SELECTION-SCREEN .

IF p_jahr LT 1 OR p_jahr GT 9999 .

MESSAGE 'Eingegenes Jahr soll zwischen 0001 und 9999 sein' TYPE 'E'.

ENDIF.

START-OF-SELECTION.

Erstellen eines kurzen Konzepts zur folgenden Problemstellung: (Zahlen umrechnen)

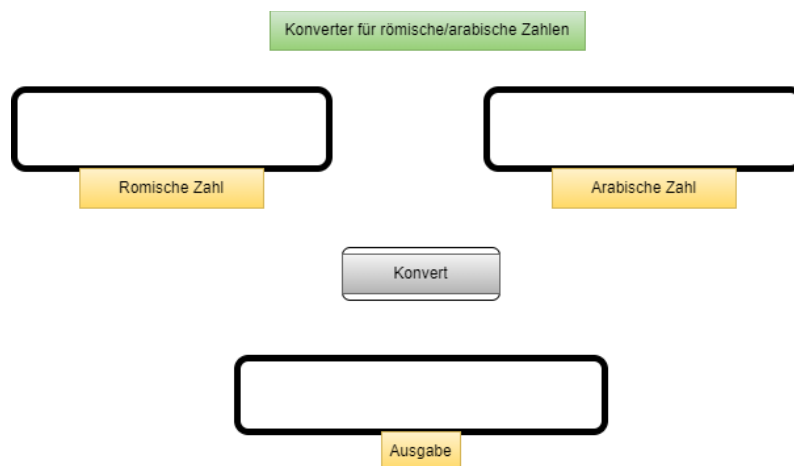
Ein Kunde möchte einen „Konverter für römische/arabische Zahlen“. Auf dem Bildschirm soll es zwei Eingabefelder geben. In ein Eingabefeld soll man arabische Zahlen eintragen können und in dem anderen sollen römische Zahlen eintragen können. Sind beide Felder gefüllt, soll es eine Fehlermeldung geben. Ist kein Feld gefüllt, soll ebenfalls eine Fehlermeldung erscheinen. Es soll geprüft werden, ob die Eingaben plausibel sind. (z.B. Eingabe von XVIII oder MF sind nicht plausibel). Je nachdem, ob das Feld arabische oder römische Zahlen gefüllt ist, soll das Programm diese konvertieren. Anschließend soll es eine Ausgabe geben. Darauf sollen die Eingabe und der konvertierte Wert sichtbar sein.

Dieses Konzept bietet ein Konzept, das römische/arabische Zahlen umgerechnet werden. Dafür hat es in 5 Schritten dargestellt. Erste ist es über Darstellung einer Benutzeroberfläche. Dann kommt es zweite Schritt als Validierung der Eingaben, d.h. die Eingabe überprüft, ob nicht nur die Felder richtig gefüllt sind, sondern die eingegebenen Eingaben plausibel sind. Anschließend wird Eingabe mit Hilfe eines Algorithmus zur Umwandlung konvertiert. Als nächstes wird die Umwandlung auf dem Ausgabebereich gezeigt. Abschließend gibt es eine Zusammenfassung und erforderliche Informationen.

1. Benutzeroberfläche:

1.1. Erstellen einer Benutzeroberfläche mit zwei Eingabefeldern, einem Konvert-Button und einem Ausgabebereich.

1.2. Das erste (links) Eingabefeld ist für römische Zahlen und das zweite (rechts) Eingabefeld für arabische Zahlen wie im folgenden dargestellt werden.



2. Validierung der Eingaben:

2.0. Mit Hilfe Input wird die Eingabe von User aufgenommen werden.

2.1. Überprüfen, ob beide Felder entweder gefüllt oder leer sind.

Mit Hilfe wenn Bedingungen kann Programm überprüfen, ob die entsprechende Felder entweder gefüllt oder leer sind. Zum Beispiel

```
wenn (Beide Eingabe Felder sind leer)) {  
    // Die Felder sind leer  
} wenn (Beide Eingabe Felder sind gefüllt) {  
    // Die Felder sind gefüllt  
}
```

2.2. Wenn beide Felder gefüllt sind oder beide leer sind, wird eine Fehlermeldung angezeigt.

```
Wenn (Beide Eingabe Felder sind leer) {  
    // print(Beide Felder sind leer.)  
    // print(Nur eine Feld soll gefüllt sein) }
```

```
Wenn (Beide Eingabe Felder sind gefüllt) {  
    // print(Beide Felder sind gefüllt.)  
    // print(Nur eine Feld soll gefüllt sein) }
```

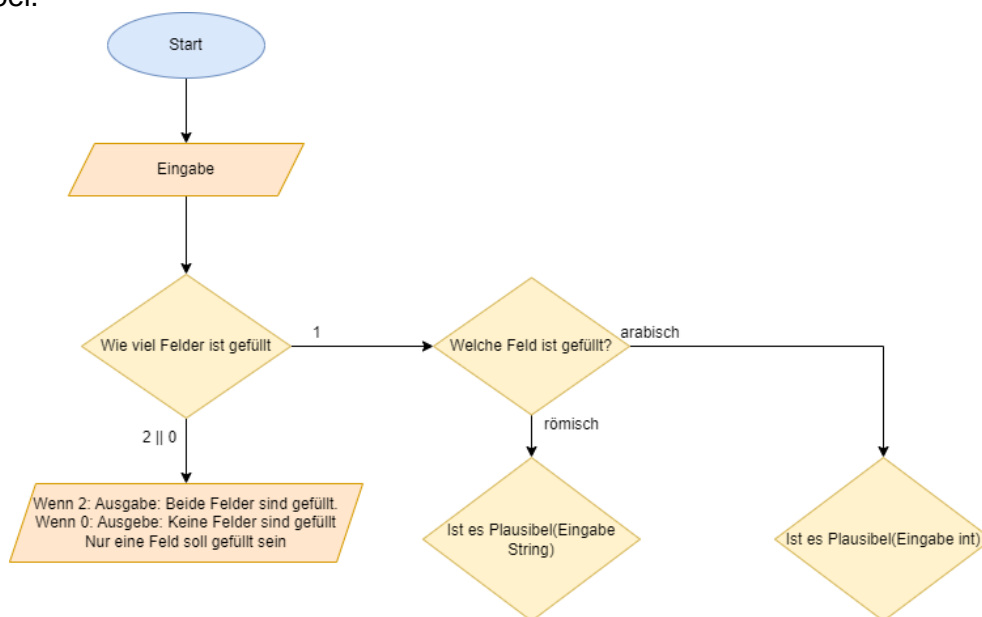
2.3. Prüfen Sie auch die Plausibilität der eingegebenen Werte (z.B. ungültige römische Zeichenkombinationen).

Mit Hilfe for oder while Schleife soll die eingegebene römische Zahl auf gesamte Länge kontrolliert werden.

Mit Hilfe von wenn Schleife soll die Ziffern der eingegebenen römische Zahl kontrolliert werden, ob alle Ziffern laut der Römische Ziffern Tabelle richtig sind.

Während der Schleife mit Hilfe wenn Bedingungen überprüft werden,
ob die Ziffer mit dem höchsten Wert steht zuerst,
ob nicht mehr als drei Ziffern einer Sorte nebeneinanderstehen,
ob zwischen Ziffern kein Komma gibt.

Wenn die oben genannte Bedingungen erfüllt sind, ist die eingegebene römische Zahl plausibel.



2.4. Römische Zahlen – Regeln¹⁷

Römische Ziffern

Großbuchstaben	I	V	X	L	C	D	M
Wert	1	5	10	50	100	500	1000

„Römische Zahlen werden nach folgenden Regeln gebildet:

- Die Ziffer mit dem höchsten Wert steht zuerst, also ganz links.
- Um höhere Zahlen zu erhalten, fügen Sie weitere Ziffern hinzu.
- Dabei werden die Werte der Ziffern addiert.

So ergibt sich der Wert der gesamten Zahl:

$$7 = VII$$

$$V + I + I$$

$$17 = XVII$$

$$X + V + I + I$$

Da es keine Ziffer für die 7 gibt, wird sie aus der V (fünf) und zwei I (eins) zusammengerechnet. Das gleiche gilt für die Zahl 17 aus dem Beispiel. Es wird immer die höchstmögliche Ziffer verwendet, hier X (zehn) und der Rest dazu addiert. Das römische Zahlssystem wird deshalb auch als Additions- bzw. Subtraktionssystem bezeichnet.“

2.5. Arabische Zahlen – Regeln¹⁸

„Arabische Zahlen funktionieren als Dezimalzahlen. Das bedeutet, dass sie in Vor- und Nachkommastellen gegliedert werden können. Die Positionen vor dem Komma haben feste Bedeutungen: Einer, Zehner, Hunderter, Tausender usw. Sobald eine Stelle die 9 erreicht hat, wird danach die nächste Position erhöht. Diese steht immer links:

37, 38, 39, 40, 41

Bei römischen Zahlen gibt es kein Komma. Die Positionen der Ziffern sind nicht in Einer, Zehner, Hunderter usw. gegliedert. Deshalb besitzt beispielsweise die Zahl hundert (C) weniger Ziffern als die Zahl acht (VIII).“

3. Konvertierung:

3.1. Wenn das römische Zahleneingabefeld gefüllt ist, wird die Zahl in eine arabische Zahl mit Hilfe eines Algorithmus zur Umwandlung von römischen in arabischen Zahlen konvertiert und im Ausgabebereich ausgegeben.

Mit Hilfe for oder while Schleife soll die eingegebene römische Zahl auf gesamte Länge gerechnet werden. Dafür deklariert eine Integer, Ergebnis zu aufzuhalten.

Während der Schleife mit Hilfe wenn Bedingungen überprüft und damit gerechnet werden,

Die Zahlen werden von links nach rechts gelesen und addiert,
I, X oder C links von einer höheren Ziffer steht, wird sie subtrahiert.

¹⁷ <https://www.studysmarter.de/schule/latein/roemische-zahlen/>

¹⁸ <https://www.studysmarter.de/schule/latein/roemische-zahlen/>

Mit Hilfe ein Switch Case und for/while Schleife o.ä. kann die Umrechnung erledigt werden. Zum Beispiel;

```
switch(X){
    case 'I' : return 1;
    case 'V' : return 5;
```

Wenn die oben genannte Schleife erfüllt ist, wird die eingegebene römische Zahl in arabische Zahl gezeigt.

3.2. Wenn das arabische Zahleneingabefeld gefüllt ist, wird die Zahl in eine römische Zahl mit Hilfe einen Algorithmus zur Umwandlung von arabischen in römischen Zahlen konvertiert und im Ausgabebereich ausgegeben.

Mit Hilfe for oder while Schleife soll die eingegebene arabische Zahl gerechnet werden. Dafür deklariert eine String, Ergebnis zu aufzuhalten.

Erstelle eine Liste von möglichen römischen Ziffern und ihren entsprechenden

Werten:

- M = 1000
- D = 500
- C = 100
- L = 50

Römische Ziffern

Großbuchstaben	I	II	III	IV	V	VI	VII	VIII	IX	X
Wert	1	2	3	4	5	6	7	8	9	10

Während der Schleife mit Hilfe wenn Bedingungen überprüft werden,

Beginnen mit der größten römischen Ziffer (M) und prüfen, wie oft diese in die eingegebene arabische Zahl gibt. Sofern es gibt, dieser Wert wird von links nach rechts geschrieben. Danach ziehen den Wert der Römischen Ziffer von der eingegebenen Zahl ab. Danach macht diese nacheinander für den Werten 500, 100, 50, 10. Unter 10 soll der Wert von der Tabelle unmittelbar abgeschrieben werden.

Wenn die Rest 0 ist, Bedingungen ist damit fertig.

Zum Beispiel;

Umrechnen (Ohne Bruch)	Als römische Zahl	Darstellung in römische Zahl	Bleibt noch übrig
			2514
Quotient von 2517/1000=2	→ MM	MM	514
514/500=1	→ D	MMD	14
14/10=1	→ X	MMDX	4
4	→ IV	MMDXIV	0

Zeigt die String.

• 3.3 Römische Zahlen umrechnen¹⁹

„Hinweis für die Umrechnung römischer Zahlen: Es gibt kein Zeichen für die 0. Die Zahlen werden von links nach rechts gelesen und addiert, wobei der Wert der Zahlzeichen von links nach rechts abnimmt. Wenn arabische in Römische Zahlen umgerechnet werden, gilt es zu beachten, dass maximal drei gleiche Symbole hintereinanderstehen können.

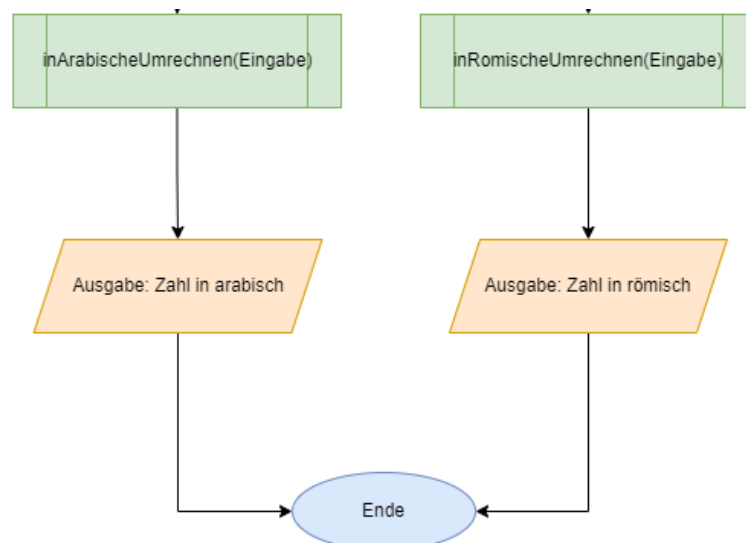
Dementsprechend wird die 5 nicht durch fünf Striche dargestellt, sondern durch V. Die untenstehenden Tabellen zeigen auf, wann statt einer Symbol-Wiederholung eine andere

¹⁹ <https://www.roemische-zahlen.net/>

Römische Ziffer benutzen werden kann. Eine 10 wird zum Beispiel nicht durch VV dargestellt, sondern durch X.

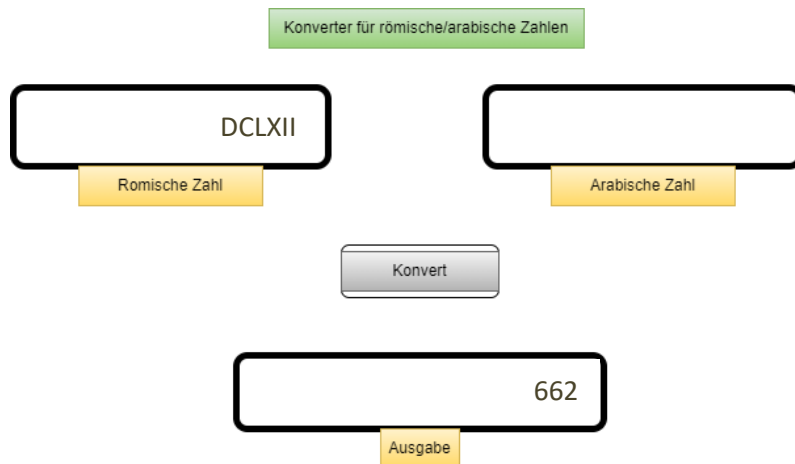
Römische Zahlen - Das Wichtigste

- Das römische Zahlensystem besteht aus den Ziffern I, V, X, L, C, D und M sowie aus zwei Arten von Zahlwörtern:
 - Grundzahlen (Cardinalia): Anzahl
 - Ordnungszahlen (Ordinalia): Reihenfolge
- Römische und arabische Zahlen – Unterschiede:
 - Die römischen Zahlen folgen nicht dem Dezimalsystem.
 - Die Werte der Ziffern werden addiert.
- Römische Zahlen – Regeln:
 - Es dürfen nicht mehr als drei Ziffern einer Sorte nebeneinanderstehen.
 - Um dies zu vermeiden, gibt es die Subtraktionsregel.
 - Wenn I, X oder C links von einer höheren Ziffer steht, wird sie subtrahiert.
- Römische Zahlen 1-10: I, II, III, IV, V, VI, VII, VIII, IX, X
 - Römische Zahlen 1-100: Die römischen Zahlen über 10 werden zusätzlich mit der Ziffer L (50) und C (100) gebildet.
 - Römische Zahlen 1-1000: Die römischen Zahlen über 100 werden zusätzlich mit den Ziffern D (500) und M (1000) gebildet.
- Römische Zahlen Datum: XXIII. IX. LXIII (23.09.63 v. Chr., Geburtsdatum des Kaiser Augustus)
 - Römische Zahlen Jahreszahl 2021: MMXXI.“



4. Anzeige der Ergebnisse:

- Nach erfolgreicher Konvertierung wird das Ergebnis auf dem Ausgabebereich gezeigt.



5. Implementierungsdetails:

- 5.1. Verwenden einer Programmiersprache je nach der Wahl (z.B., Python, Java).
- 5.2. Verwenden von geeigneten Funktionen oder Algorithmen zur Umwandlung von römischen in arabische Zahlen und umgekehrt.
- 5.3. Verwenden Sie vielleicht geeignete Validierungsmechanismen, um sicherzustellen, dass die Eingaben korrekt sind.

