PROJEKTDOKUMENTATION

Abschlussprojekt Winter 2024/2025

Fachinformatiker für Anwendungsentwicklung

THEMA

Digitale Bereitstellung von Produktionslaufkarten und zugehörigen Dokumenten auf mobilen Arbeitsgeräten

Prüfungsteilnehmer:	Daniel Malychko [redacted]
Praktikumsbetrieb:	IAP GmbH Valentinskamp 30 20355 Hamburg
Projektbetreuerin:	Hanna Balski
Durchführungszeitraum:	23.10.2024 – 11.12.2024

Inhalt

GLOSSAR

OpenEdge ABL ist eine datenbankbasierte Programmiersprache mit dem Fokus auf die betriebliche Verwaltung

Legacy Support ist die Unterstützung für veraltete Systeme, die für den Betrieb noch benötigt werden

Enterprise-Resource-Planning-Systeme sind Softwarelösungen, welche zur Verwaltung der geschäftlichen Ressourcen wie Personal, Material, Kapital, Arbeit dienen

OF-1 ist das Low Code basierte Framework der IAP

User Interface bezieht sich auf die Schnittstelle, über die Benutzer/innen mit der Software oder Hardware interagieren

Bei einer UI-Maske handelt es sich in OF-1 um ein auf einen Anwendungsfall zugeschnittenes Unterprogramm

.NET: eine von Microsoft entwickelte Entwicklungsplattform für Anwendungsentwicklung auf Microsoft Betriebssystemen, welche eine Sammlung von Frameworks enthält

Ein Add-On ist eine Softwareerweiterung, die Funktionen zu Anwendungen hinzufügt

PCase: Programm zur Modellierung von OpenEdge Datenbanken

TortoiseSVN: Die Software ermöglicht den Abgleich mit dem Versionsverwaltungssystem und damit ein gemeinsames Arbeiten an Projekten

Komponenten auf dem OF-1 Screen: Elemente, die OF-1 zum Erzeugen von Screens bereitstellt

Hooks: Events von OF-1 Komponenten. Zum Beispiel beim Klicken eines Buttons oder Speichern eines Datensatzes aus einem Container

Zoom-Komponente: Eine Suchkomponente, die auf eine andere Datenbanktabelle schaut. Statt des tatsächlichen Feldinhalts (Fremdschlüssel) lässt sich ein beliebiges anderes Feld der fremden Tabelle anzeigen und zur Auswahl eine Liste mit Suchfunktion öffnen

Mandanten abhängig: Ein Mandant ist im Kontext zu proAlpha ein Schlüssel in der Datenbank, welcher die Steuerung von zum Beispiel verschiedensten Werken der Firmen ermöglicht.

Relationsstruktur: Die Beziehungsstruktur, wie Daten in der proALPHA Datenbank miteinander verknüpft sind.

Code Review: Arbeitsmethode. um Quell-Code auf Qualität zu prüfen. Hierbei wird der Quell-Code manuell von einer anderen Person geprüft

1 Einleitung

Die folgende Projektdokumentation schildert den Ablauf und Aufbau des IHK-Abschlussprojektes **„Digitale Bereitstellung von Produktionslaufkarten und zugehörigen Dokumenten auf mobilen Arbeitsgeräten.“**

1.1 Projektumfeld

Die IAP GmbH ist seit 1992 ein zuverlässiger Partner bei der Beratung und Lösung technischer Herausforderungen. Der Schwerpunkt des Unternehmens liegt auf der Entwicklung von Geschäftsanwendungen mit der OpenEdge Advanced Business Language von Progress, wobei die IAP mittlerweile zu den führenden Partnern in Europa zählt. Mit rund 45 Mitarbeitern, die überwiegend in Hamburg ansässig sind, bietet IAP sowohl internen als auch externen Support.

Das Hauptklientel besteht aus mittelständischen Unternehmen, welche Unterstützung im Bereich Legacy Support benötigen oder eine Migration bzw. Neuentwicklung von ERP-Systemen auf Basis von OpenEdge anstreben. Dabei kommt das eigens entwickelte OF-1 Framework zum Einsatz, das maßgeschneiderte Lösungen ermöglicht.

1.2 Projektziel

Ziel des Projekts ist es, die papierbasierten Dokumente durch ein digitales System zu ersetzen und als Add-on auf den Arbeitsgeräten im Betrieb zur Verfügung zu stellen. Das Integrieren einer digitalen Laufkarte mit den dazugehörigen Dokumenten soll sicherstellen, dass diese jederzeit und in der korrekten Reihenfolge abrufbar sind. Aufgrund dessen wird für jedes aufgeführte Dokument eine individuelle OF-1 User Interface Maske erstellt, in der die entsprechende Logik vorhanden ist. Die Laufkarte wird als Hauptmaske dargestellt, während die dazugehörigen Dokumente als Unteransichten in anklickbaren Reitern bzw. Tabs angezeigt werden.

1.3 Projektbegründung

Der aktuelle Prozess sieht vor, dass Laufkarten und die dazugehörigen Dokumente ausgedruckt und manuell sortiert werden müssen. Dieser papierbasierte Ansatz ist zeitaufwendig, fehleranfällig und erzeugt unnötigen Papierverbrauch. Zudem besteht das Risiko, dass wichtige Dokumente verloren gehen oder falsch sortiert werden, was zu Verzögerungen im Produktionsprozess führt.

1.4 Projektzeitraum

Der Projektzeitraum erstreckt sich vom 23. Oktober bis zum 11. Dezember und umfasst insgesamt 80 Stunden.

1.5 Abweichungen zum Projektantrag

Im Projektantrag wurden Feier-, Brücken- und Schultage für den Projektzeitraum nicht berücksichtigt. Der Zeitplan und die geplanten Projektschritte wurden weitgehend eingehalten.

1.6 Lesehinweise

Fachbegriffe werden in dem Glossar erklärt, um die Lesbarkeit und das Verständnis der Dokumentation zu erleichtern.

2 Planungsphase

2.1 Zeitplanung

Die Anforderungen an die Projektdokumentation erlauben nicht wesentlich weniger oder mehr als 80 Stunden für die Projektdurchführung. Da einige Prozesse nicht voneinander abhängig sind, wurde zur besseren Planung ein modifizierter Netzplan erstellt. Durch die Festlegung auf 80 Stunden und nur einen Projektteilnehmer entfällt der Freie Puffer. Der früheste Anfangszeitpunkt eines Prozesses berechnet sich bei parallelen Prozessen aus den aufsummierten Prozessdauern.

Zeitliche Abweichungen der einzelnen Phasen werden im Kapitel 5.2 Soll-Ist Zeitvergleich erläutert. Eine Übersicht der Phasen mit Gantt-Diagramm befindet sich im Anhang C.

2.2 Analyse der bestehenden Dokumente

Die benötigte Laufkarte befindet sich innerhalb des proAlpha ERP-Systems und steht mit einer vorhandenen Funktion zum Drucken zur Verfügung. Zusätzlich zu den bestehenden Laufkarten sollen die dazugehörigen Zeichnungen und Produktionsdatenblätter hinzugefügt werden. Im Rahmen dieses Projekts wurden die Laufkarten und die Zeichnungen implementiert. Die Produktionsdatenblätter hätten den Projektzeitraum überschritten.

2.2.1 Erarbeiten der Datenfelder pro Dokument

Für Laufkarten sollen die zugehörigen Zeichnungen, die Prozessdatenblätter und alle in der Laufkarte zugehörigen Datenfelder aufgenommen werden. Für Zeichnungen wird die Teilenummer, die technischen Zeichnungen und der Mandant benötigt. Bei den Prozessdatenblättern wird die Auftragsnummer der jeweiligen Laufkarte benötigt.

2.3 IST Analyse

Der aktuelle Stand des Produktionsprozesses erfordert eine manuelle Verwaltung der Laufkarten und der zugehörigen Dokumente. Die Laufkarte(n) müssen im proAlpha System unter dem Programm „Produktionsauftrag“ ausgedruckt werden, die Zeichnung(en) sind im Programm „Standardstücklisten“ im Menü „Funktion“ und dort unter „Zeichnungen“ zu finden, die Prozessdokumente sind unter „Dokumente“ in einer Unterdokumentart der Teiledokumente abgelegt und können dort ausgedruckt werden. Abbildungen dieser Bildschirme finden sich in den Anhängen
E.a. Laufkartenmaske, E.b.Zeichnungsmaske und E.c.Prozessdatenblattmaske.

2.4 SOLL Konzept

Ein klassisches Lasten- oder Pflichtenheft existiert für dieses Projekt nicht. Die grundlegenden Projektkonzepte wurden in Fachgesprächen ermittelt und festgehalten.

2.4.1 Muss-Kriterien

- Anlegen von Masken für Laufkarte, zugehörige Zeichnungen, zugehörige Prozessdatenblätter.
- Anzeigen der Masken in der Reihenfolge: Laufkarte - zugehörige Zeichnung(en) - zugehörige Prozessdatenblätter.
- Mandantenabhängige Steuerung der Daten in der Zeichnungsmaske.
- Entscheidbarkeit auf Artikelebene durch ein Eingabefeld in der Zeichnungsmaske, für welches Teil die Folgedokumente angezeigt werden.

2.4.2 Kann-Kriterien

- Jedes Dokument ist separat über eine eigene UI-Maske zugeordnet
- Ausfüllen relevanter Datenfelder erfolgt automatisch mithilfe der Datenbankanbindung und Methoden

2.5 Make or Buy-Entscheidung

2.5.1 Einkauf eines Fremdproduktes

OF-1 ist eine Software, die von Mitarbeitern der IAP GmbH entwickelt und an Kunden vertrieben wird. Ein Hauptbestandteil von OF-1 ist der visuelle Designer. Mit diesen Technologien in Kombination, bietet sich die Möglichkeit Add-ons für bestehende ERP-Software wie von proAlpha zu erzeugen. Eine externe Beauftragung für ein Fremdprodukt mit anderen Technologien als OF-1 würde keine Vorteile bringen. Im Gegenteil würde sie zu erheblichen Mehrkosten führen. Außerdem wäre keine Zeitersparnis zu erwarten. Diesem stehen die Kosten für das Add-on mit OF-1 gegenüber, das erst entwickelt werden muss.

Bei der Betrachtung eines Fremdproduktes sind nicht nur die Entwicklungskosten zu berücksichtigen, sondern auch mögliche langfristige Aufwendungen für Lizenzen und Support. Diese Faktoren machen die Option eines Fremdproduktes sowohl aus finanzieller als auch aus praktischer Sicht unattraktiv.

Darüber hinaus ist die Integration der entwickelten Software in proAlpha zu berücksichtigen. Dies kostet Zeit und schränkt die Auswahl der möglichen Angebote ein.

OpenEdge bietet die Möglichkeit .NET-Elemente zu verwenden, wodurch in OF-1 leere Screens erstellt werden können, die dann über die Quellcodeebene in das Fremdprodukt eingebunden werden können.

Die allgemeine Anpassung des Fremdprogramms ist auf die vom Hersteller gegebenen Möglichkeiten beschränkt. Außerdem muss eine Schnittstelle zwischen OF-1 und der proAlpha Datenbank geschaffen und die bestehenden Daten aus proAlpha in das neue OF-1 Programm migriert werden.

2.5.2 Gegenüberstellung

Die möglichen Lösungen werden mithilfe einer Vergleichsmatrix gegenübergestellt. In dieser werden sowohl funktionale als auch preisliche Punkte berücksichtigt.

Die anfallenden Kosten, bestehend aus aufgebrachter Arbeitszeit und Fremdkosten, erhalten insgesamt den Faktor “50”. Die funktionalen Punkte werden auch insgesamt mit “50” gewichtet, wodurch eine ausgeglichene Entscheidungsfindung ermöglicht werden soll.

		Erweiterung des bestehenden Systems		Einkauf eines Fremdproduktes	
Kriterium	**Faktor**	**Wert**	**Gew. Wert**	**Wert**	**Gew. Wert**
Fremdkosten	25	100	25	30	7,5
Eigenkosten (Zeitaufwand)	25	30	7,5	80	20
Performance	20	100	20	70	14
Design	20	100	20	50	10
Zukunftssicherheit	10	80	8	70	7
Ergebnis		80,5		58,5	

Tabelle 1: Vergleich der Umsetzungswege

2.5.3 Begründung der Entscheidung

Die Erweiterung des bestehenden Systems bietet im Vergleich zu einer externen Beauftragung deutliche Vorteile bei den Kosten und auch bei den qualitativen Kriterien, wie der Performance und dem von OF-1 gewohnten Design, wodurch sich die Mitarbeiter schneller zurechtfinden. Die Dokumente müssen anschließend nicht mehr manuell ausgedruckt und sortiert werden, was häufig zu Verzögerungen, Fehlern und erhöhtem Papierverbrauch geführt hat.

Daher wurde sich für die Weiterentwicklung eines Add-ons mit OF-1 entschieden.

2.6 Wirtschaftlichkeit

2.6.1 Projektkosten

Für die Kosten werden Personalkosten herangezogen. Auch wenn es sich um eine Umschulung handelt, wird bei den eigenen Kosten von einem Auszubildenden ausgegangen. Laut dem Personalwesen liegen die Stundenkosten eines Auszubildenden bei 6 € und eines Seniors bei 80 €. Die Stundenkosten der Ansprechpartner liegen bei 22 €. Auf die Lohnkosten werden nochmal 15 % Fertigungsgemeinkosten (z. B. Miete, Strom, Lizenzen, Schreibmaterial und Hardware) und 5 % Verwaltungsgemeinkosten (z. B. Lohnbuchhaltung) aufgerechnet

80h * 6 €/h = 480 €
6h * 22 €/h = 132 €
5h * 80 €/h = 400 €
480 € + 132 € + 400 € = 1012 €
1012 € + (1012 € * 0,15) + (1012 € * 0,05) = 1214,40 €

Tabelle 2: Projektkosten Rechnung

Die Gesamtkosten für das Projekt belaufen sich auf 1214,40 €.

2.6.2 Rentabilität

Mit dem Add-On für das proAlpha ERP entstehen Zeitersparnisse bei Produktionsaufgaben. Dies geschieht durch den schnelleren Zugriff auf Dokumente und auch die automatisierte Sortierung mit den in den Eingabemasken angezeigten Dokumenten.

Bei einer Zeitersparnis von 5 Minuten pro digitalisierten Dokumenten, 15 Dokumentenverwendungen am Tag, 20 Arbeitstagen im Monat und 45 € Personalkosten pro Stunde ergibt sich eine Amortisation von um die 1,08 Monate → gerundet 2 Monate.

$Zeitersparnis\ pro\ Monat: 5min \times 15 \times 20 = 1500min = 25h$
Geldeinsparungen pro Monat: 25 h × 45 € = 1125 €
Amortisation: 1214,40 / 1125 ≈ 1,08 Monate
gerundet 2 Monate

Tabelle 3: Amortisationsrechnung

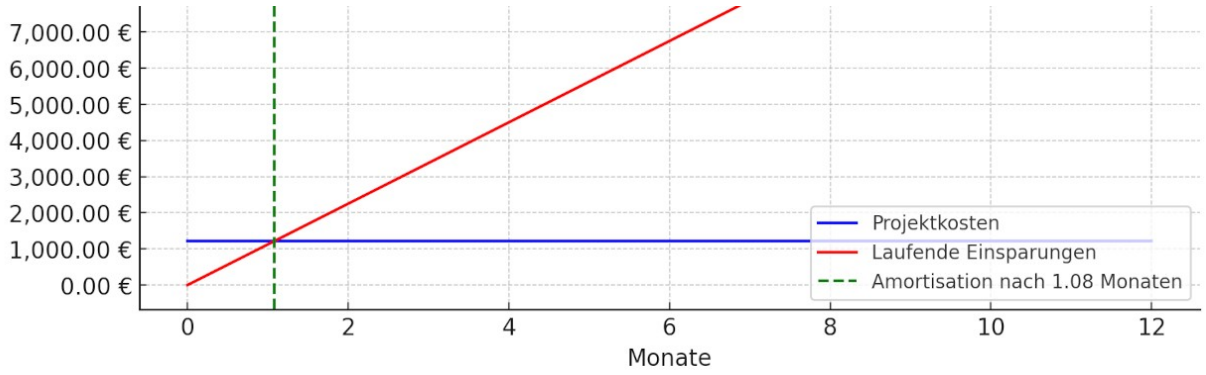


Abbildung 1: Rentabilität

2.6.3 Nutzwertanalyse

Kriterium	Gewichtung (Punkte)	Digitale Bereitstellung	Punktzahl (digital)	Gedruckte Bereitstellung	Punktzahl (gedruckt)
Kosten	10	7	70	5	50
Zeitaufwand für das Unternehmen	15	9	135	6	90
Kundenfreundlichkeit (Usability)	20	9	180	6	120
Fehleranfälligkeit	15	9	135	5	75
Zugänglichkeit der Informationen	15	10	150	6	90
Bedienbarkeit im Unternehmen	10	9	90	7	70
Reduktion des Papierverbrauchs	15	10	150	3	45
Bewertung (10 sehr gut, 1 sehr schlecht) Gesamtpunktzahl: 910 (digital) gegenüber 540 (gedruckt).					

Tabelle 4: Nutzwertanalyse

Kosten (10 Punkte):

Die digitale Lösung hat anfängliche Implementierungskosten, während die gedruckte Lösung laufende Material- und Arbeitskosten verursacht. Daher erhält die digitale Lösung eine höhere Bewertung (7 gegenüber 5).

Zeitaufwand für das Unternehmen (15 Punkte):

Die digitale Bereitstellung reduziert den Zeitaufwand für das Suchen und Sortieren von Dokumenten erheblich und erhält daher eine höhere Bewertung (9) als die gedruckte Version (6).

Benutzerfreundlichkeit (Nutzbarkeit) (20 Punkte):

Die digitale Lösung ermöglicht einen schnelleren Zugriff und eine flexiblere Nutzung durch die Mitarbeiter, was die Benutzerfreundlichkeit im Vergleich zur gedruckten Bereitstellung deutlich verbessert (9 gegenüber 6).

Fehleranfälligkeit (15 Punkte):

Durch die Digitalisierung werden manuelle Fehler reduziert, was die Fehleranfälligkeit der gedruckten Version erhöht. Hier schneidet die digitale Lösung besser ab (9 zu 5).

Zugänglichkeit von Informationen (15 Punkte):

Die digitale Bereitstellung macht die Informationen überall und jederzeit zugänglich. Im Gegensatz dazu erfordert die gedruckte Version eine physische Präsenz, weshalb die digitale Lösung deutlich höher bewertet wird (10 gegenüber 6).

Benutzerfreundlichkeit im Unternehmen (10 Punkte):

Beide Lösungen sind gut zu handhaben, wobei die digitale Lösung aufgrund ihrer benutzerfreundlichen Oberfläche eine einfachere Handhabung ermöglicht. Bewertung: 9 (digital) gegenüber 7 (gedruckt).

Verringerung des Papierverbrauchs (15 Punkte):

Die digitale Lösung spart Papier vollständig ein und erreicht die maximale Punktzahl von 10, während die gedruckte Lösung Papier verbraucht und mit 3 Punkten bewertet wird.

Die Gesamtpunktzahl von 910 für die digitale Bereitstellung im Vergleich zu 540 für die gedruckte Variante zeigt, dass die digitale Lösung klare Vorteile bietet. Durch die höhere Effizienz, die geringere Fehlerquote und die positiven Auswirkungen auf den Papierverbrauch ist die digitale Bereitstellung nicht nur wirtschaftlicher, sondern auch nachhaltiger und benutzerfreundlicher als die gedruckte Bereitstellung.

2.7 Ressourcenplanung

Das gesamte Projekt wird am Arbeitsplatz durchgeführt. Dort stehen alle notwendigen Soft- und Hardware zur Verfügung.

Hardware:

- Intel NUC Kit NUC6i5SYB
- Maus
- Tastatur
- Flachbildschirm

Software:

- Lokale virtuelle Maschine (Datenbankanbindung und proAlpha)
- Microsoft Windows 10 Pro
- LibreOffice
- Eclipse mit Plugins für OpenEdge ABL, OF-1 und PCase als Entwicklungsumgebung
- TortoiseSVN

Verwendete Programmiersprachen:

- OpenEdge ABL

3 Implementierungsphase

3.1 Erstellen der Eingabemasken

Die Masken werden vom Framework in Form von .ofx-Dateien gespeichert, die eine XML-Struktur aufweisen. Bei der späteren Ausführung der Programme werden sie über eine Abstraktionsebene, die sogenannten "W-Klassen", dargestellt. Dabei werden die Inhalte entweder in .NET oder im Skin-Client für HTML5 und Java angezeigt. Die Masken wurden mithilfe des OF-1 Designers entwickelt. Dieser ermöglicht eine effiziente und benutzerfreundliche Erstellung von Oberflächen, ohne dass Frames manuell codiert werden müssen. Darüber hinaus bietet der OF-1 Designer eine integrierte Schnittstelle zur Datenbank, was die Entwicklung weiter vereinfacht.

3.1.1 Gestaltung im Designer

Die Positionierung von Komponenten auf dem OF-1 Screen sowie deren Anpassung über Eigenschaftsmenüs erfolgt durch den visuellen Designer. Die Komponenten umfassen eine Vielzahl von Elementen, darunter Container, die eine Beziehung zu Datenbanktabellen darstellen, sowie Steuerelemente wie Buttons, Fill-Ins, Text und so weiter. Über angewendete Hooks an den erwähnten Komponenten lassen sich durch Code benutzerdefinierte Aktionen ausführen. Der Code wird in den Screen-Klassen gespeichert. Hierbei gibt es immer eine "-c-g"-Klasse für den von OF-1 automatisch generierten Code. Diese vererbt die Komponenten-Objekte an die "-c"-Klasse, in denen die Entwickler ihren Code schreiben können.

Zur Gestaltung der Auswahl verschiedener Dokumentenarten werden Container hinzugefügt. Diese bieten die Möglichkeit, mehrere Bereiche zu erstellen, zudem können in dem Container Text, Fill-In, Rectangle-Felder und vieles mehr integriert werden. Über die oCurrent-Property kann zudem der aktive Container zur Anzeige und Ausführung von Hooks anders eingestellt werden. Um eine Vater-Kind-Beziehung von Container zu Container herzustellen, bieten die Container auch die Möglichkeit, über Properties miteinander verlinkt zu werden. Dies ermöglicht, dass Daten aus den Business Entity Schleifen effizient vom Vater-Container an den Kind-Container übertragen werden, insbesondere über Fill-Ins in Kombination mit Search Properties und einem Custom-Browse.

3.1.2 Laufkarten Maske

Die Container mit Darstellung der Laufkarte/n werden in solche Container umgewandelt. Da der Datenbankzugang wegen der besonderen Gestaltung des Verhältnisses über OD_DocRel schon über Hooks beim Anzeigen und Speichern von Datensätzen erfolgt, muss dieser nicht verändert werden. Die Laufkartenmaske wurde in drei Container unterteilt: LaufkartenContainer (Parent): Dieser

Container bildet das Grundgerüst der Maske und enthält verschiedene OF-1 Komponenten: ein Rectangle mit einem Fill-In für die dynamische Eingabe der Rückmeldenummer. Eine Bildkomponente für das Logo, Textfelder für wichtige Informationen wie Titel, Auftragsnummer, Endprodukt, Menge, Priorität, Prozess, Erstellungsdatum, Start- und Endtermin sowie Lagerort. Zwei Buttons für die Navigation zur Zeichnungsmaske und zur Prozessdokumentenmaske.

Die Verwendung von Containern und Accordions ermöglicht eine übersichtliche und strukturierte Darstellung der komplexen Informationen auf der Laufkarte. Die Möglichkeit zum Auf- und Zuklappen durch das Accordion Objekt von Bereichen erhöht die Benutzerfreundlichkeit und erleichtert die Navigation innerhalb der Maske. Custom-Browser in den Material- und Arbeitsfolgen-Containern bieten eine flexible Lösung zur Anzeige und Verwaltung der dynamischen Daten. Sie ermöglichen eine effiziente Darstellung wiederkehrender Informationen und erleichtern die Interaktion mit den Datensätzen.

Durch den Einsatz des OF-1 Designers und seiner Komponenten konnte eine ansprechende und funktionale Benutzeroberfläche geschaffen werden, die alle Anforderungen an die Laufkartenmaske erfüllt. Die integrierte Datenbankanbindung und die vorgefertigten Komponenten des OF-1 Frameworks trugen wesentlich zur effizienten Entwicklung bei.

MaterialContainer (Child von LaufkartenContainer): Dieser Container enthält ein Accordion mit einer Accordion-Page, über die der Inhalt ein- und ausgeklappt werden kann. Innerhalb der Accordion-Page ist ein Custom-Browse implementiert, der für die Darstellung wiederkehrender Daten und Abfragen konzipiert ist. Der Custom-Browse enthält Textkomponenten für Artikel, Bezeichnung, Bedarfsmenge, Mengeneinheit und Barcode sowie eine Bildkomponente zur Visualisierung des Barcodes.

ArbeitsFolgenContainer: Dieser Container wird für das Accordion mit einer Accordion-Page benötigt. Innerhalb dieser Seite befindet sich ein weiterer Container namens AccordionContainerFolgen (ebenfalls ein Child von LaufkartenContainer). Dieser enthält einen Custom-Browse mit umfangreichen Textkomponenten wie Aktivität, AktivitätenText, RückMeldeKz, Rückmeldung, BasisRessource, BasisRessourceText, SollrüstZeit, rüstZeitEinheit, SollStückZeit, StückZeitEinheit, SollZeitGesamt, ZeitGesamtEinheit, StartTerminAkt, EndTerminAkt und BarcodeCodeArbeitsFolgen. Zusätzlich wurde eine Bildkomponente für den Barcode integriert.

3.1.3 Generierung der Barcodes

Um die dynamisch generierten Barcodes als PNG-Dateien in die Bildkomponenten des OF-1 Designers einzufügen und anschließend als scanbare Code-128-Barcodes darzustellen, habe ich mich für die Verwendung der API von TEC-IT entschieden. Diese API ermöglicht die Konfiguration von Barcodes durch das Hinzufügen verschiedener Parameter in die URL der Schnittstelle. Die verwendeten Parameter sind unter anderem &dpi=94, &modulewidth=0.19, &height=6, &imagetype=png, &download=true und &hidehrt=true. Durch diese Anpassungen kann ich den generierten Barcode genau nach meinen Bedürfnissen gestalten.

Um den Barcode automatisch zu speichern, verwende ich einen Shell-Befehl, der die PNG-Datei in dem angegebenen Ordner speichert. Beim Schließen der Laufkartenmaske werden die generierten Barcodes mit einem weiteren Shell-Befehl und dem BeforeReturn-Hook wieder gelöscht. Dies gewährleistet einen sauberen Umgang mit den generierten Daten und verhindert eine Anhäufung unnötiger Dateien. Zusätzlich habe ich den Code für einen der Container aus den Business Entity Klassen in einem kurzen Bildabschnitt eingefügt, um die Implementierung zu veranschaulichen. Diese Vorgehensweise ermöglicht es mir, effizient und effektiv mit den Barcodes zu arbeiten und gleichzeitig sicherzustellen, dass alle generierten Daten korrekt verwaltet werden.

```
40
41      // Generieren und Speichern des Barcode-Bildes über einen Shell Command
42      hd = hbTT:BUFFER-FIELD("BarcodeMaterialBild") NO-ERROR.
43      IF VALID-HANDLE(hd) THEN
44      DO:
45        DEFINE VARIABLE cUri          AS CHARACTER NO-UNDO.
46        DEFINE VARIABLE cFilePath     AS CHARACTER NO-UNDO.
47        DEFINE VARIABLE cCommand      AS CHARACTER NO-UNDO.
48        DEFINE VARIABLE cbarcodeValue AS CHARACTER NO-UNDO.
49        DEFINE VARIABLE lFileExists   AS LOGICAL NO-UNDO.
50
51        // Erstellen des Barcode-Wertes und der URL für die Barcode-Generierung
52        cbarcodeValue = "000" + hbDB::RueckMeldeNr + hbDB::StructCode.
53        cUri          = "https://barcode.tec-it.com/barcode.ashx?data=" + cbarcodeValue + "`&code=Code128`&translate-
  esc=off`&dpi=94`&modulewidth=0.19`&height=6`&unit=Fit`&imagetype=png`&download=true`&hidehrt=true".
54        cFilePath     = SUBSTITUTE("e:/proalpha/pa-de-9/demo/zv/pic/" + "&1", cBarcodeValue + "Barcode" + ".png").
55
56        // Überprüfen, ob die Datei bereits existiert
57        lFileExists   = SEARCH(cFilePath) <> ?.
58
59        // Wenn die Datei nicht existiert, herunterladen und speichern
60        IF NOT lFileExists THEN
61        DO:
62          cCommand = SUBSTITUTE('powershell -Command Invoke-WebRequest -Uri "&1" -OutFile "&2"', cUri, cFilePath).
63          OS-COMMAND SILENT VALUE(cCommand).
64        END.
65
66        hd:BUFFER-VALUE = cFilePath.
67      END.
```

Abbildung 2: stkZeileBarcodeGenerate

3.1.4 Design der Laufkarte

Mit OF-1 kann man ein einheitliches Design für Anwendungen einfach erstellen. Mit den Eigenschaften "x" und "y" können Komponenten pixelgenau auf der Benutzeroberfläche platziert werden. Über zentrale Eigenschaften wie Schriftart, Hintergrundfarbe, Textfarbe und Beschriftungsfarbe können Entwickler so ein professionelles Erscheinungsbild erstellen, ohne sich mit komplexen Designprozessen beschäftigen zu müssen.

Bei den Text- und Fill-In Objekten wurden folgende Style Properties gesetzt: Font = „Arial,8,bold“ BGColorRGB = „255,255,255“, FGColorRGB = „000,000,000“. Die dazugehörigen Labels haben Labeloption Properties von Color = „000,000,000“, Font = „Arial,8,bold, und Height = 13 bekommen.

Eine detaillierte Abbildung der erstellten Benutzeroberfläche für die Laufkartenmaske befindet sich im Anhang unter E. Benutzeroberfläche Laufkarte.
Ebenfalls jeweils darauf folgend die Entwicklerumgebung, also die Designer-Ansicht der Maske unter F. Designer Ansicht Laufkarte.

3.1.5 Zeichnungs- Maske

Zur Strukturierung der Oberfläche wurden drei Container verwendet, HauptContainer, ZeichnungsContainer und ViewZeichnungscontainer.

Der HauptContainer dient als übergeordneter Container und enthält ein Rectangle, in dem verschiedene Komponenten platziert wurden. Ein Bildelement für das Logo, Fill-Ins für die Eingabefelder "Teil" und "Mandant", Textfelder für "Titel" und "Beschreibung".

Der ZeichnungsContainer(child) ist mit dem HauptContainer(parent) verlinkt und enthält einen Custom Browse mit den Textobjekten: ZeichnungsNummer, Hauptzeichnung und ProduktionsRelevant. Diese Textfelder dienen der Anzeige und Bearbeitung von Metadaten zu den Zeichnungen und sind direkt mit den entsprechenden Feldern in der Datenbank verknüpft. Dadurch ist eine automatische Aktualisierung und Speicherung der Informationen gewährleistet.

Der ViewZeichnungsContainer ist der child-Container von dem ZeichnungsContainer(parent) und enthält eine Bildkomponente zur Darstellung der Zeichnungen sowie die CadNumber als Textobjekt für die Darstellung des Pfades der jeweiligen Zeichnung. Dieser Browse wurde speziell angepasst, um die Zeichnungen übersichtlich darzustellen.

Die intelligente Verknüpfung der Container ermöglicht eine dynamische Aktualisierung der Inhalte. Trifft ein Benutzer im Custom-Browse des Zeichnungscontainers eine Auswahl, so passen sich die Inhalte im ViewContainer automatisch an. Dies betrifft sowohl das Bildtextfeld in der

Zeichnungsmaske als auch die angezeigte Zeichnung mit dem Bildobjekt. Diese dynamische Funktionalität erhöht die Benutzerfreundlichkeit und Effizienz bei der Arbeit mit Zeichnungen.

Eine detaillierte Abbildung der erstellten Benutzeroberfläche für die Zeichnungsmaske befindet sich im Anhang unter G. Benutzeroberfläche Zeichnung.
Ebenfalls jeweils darauf folgend die Entwicklerumgebung, also die Designer Ansicht der Maske unter G. Designer Ansicht Zeichnung.

3.1.6 Optimierung der Laufkartenmaske durch eine Submaske

Für die Laufkartenmaske habe ich eine Submaske erstellt, die in Kombination mit der Suchfunktion der Fill-ins verwendet werden kann. Durch Setzen der SearchField-Eigenschaft auf das Rückmeldenummer-Fill-in und Auswahl der Submaske Aufträge in der SearchScreen-Eigenschaft wird eine übersichtliche Suche ermöglicht.

Sobald der Benutzer in der Benutzeroberfläche auf die Lupe neben dem Rückmeldenummer-Fill-in klickt, öffnet sich die Aufträge-Submaske. Hier kann oberhalb der Felder Rückmeldenummer, Teil, Firma und Auftrag mit den Filteroptionen gezielt nach den gewünschten Daten gesucht werden.

3.1.7 Aufbau der Aufträge-Submaske

Die Aufträge-Submaske ist in einem Container aufgebaut, der einen Browser enthält. Dieser Browser zeigt die Spalten "RückMeldeNr", "Artikel", "Firma" und "Auftrag" aus der Business Entity be-pp_Auftrag durch entsprechende Browser-Collums an. Diese übersichtliche Anordnung erleichtert die Navigation und Datenabfrage.

3.1.8 Verwendung in der Zeichnungsmaske

Die gleiche Submaske wird auch in der Zeichnungsmaske verwendet, indem sie im Fill-in-Teil mit der SearchField-Eigenschaft geöffnet wird. Dadurch kann der Benutzer auch in dieser Maske dynamisch und benutzerfreundlich nach Daten filtern und suchen.

Ein Screenshot der Submaske-Aufträge beim Aufruf in der Laufkartenmaske befindet sich im Anhang I. SubMaskeAufträge.

3.1.9 Datenbankanbindung Laufkarte

Es existieren bereits fertige proAlpha Datenbanktabellen, die im Designer über OF-1 zur Verfügung stehen. Mithilfe des "Data Dictionary" und PCASE, welche die Generierung von OpenEdge Datenbanken in einer grafischen Oberfläche ermöglicht und des proAlpha Datenbank Managers, um die Verbindungen zwischen den für den Inhalt der Laufkarte benötigten Entitäten zu finden, wurden OF-1 Business Entities erstellt. In der Oberfläche können Tabellen, Felder und Indizes definiert werden.

Die Anzeige einiger Daten auf der Laufkarten- und Zeichnungsmaske muss manuell auf Backend Server Seite (in den Business Entity Klassen) programmiert werden. Der Code ist als Screenshot unter den entsprechenden Datenbankmodellen aufgelistet. Diese vereinfachten Datenbankmodelle der verwendeten Datenbanktabellen sind im Anhang unter D.Datenbankmodell zu finden.

3.1.10 Calculated Fields Business Entity

OF-1 bietet bei der Erstellung der Business Entitys die Möglichkeit Calculated Fields zu erstellen, welche dann in der entsprechenden .cls klasse der Business Entity angesprochen wird und somit nach Belieben angepasst werden kann.

3.1.11 Calculated Fields be-PP_Auftrag

Es wurden insgesamt für die be-PP_Auftrag die Felder "Bezeichnung", "EndproduktBezeichnung", "MengeUndEinheit", "GeoeffnetAm", "PrioritaetUndBez" und "LagerOrtUndBez" hinzugefügt.

3.1.11.1 Die Logik für das Feld Bezeichnung

Es wird ein Datensatz aus der Tabelle S_ArtikelSpr gesucht, der den aktuellen Artikel, die deutsche Sprache und eine spezifische Firma beinhaltet. Aus diesem Datensatz werden die vier Bezeichnungsfelder der Arrays (Bezeichnung[1] bis Bezeichnung[4]) extrahiert. Die Arrays in OpenEdge fangen bei 1 an, anstatt wie bei anderen Programmiersprachen bei 0. Diese Werte werden dann mit Zeilenumbrüchen (~n) zu einem neuen Array zusammengefügt. Der resultierende Array wird im Feld Bezeichnung[1] bis Bezeichnung[4] der temporären Tabelle gespeichert. Sollte kein passender Datensatz gefunden werden oder ein Fehler auftreten, wird stattdessen der Standardwert "X" verwendet.

```
// Verarbeitung des Feldes "Bezeichnung"
hd = hbTT:BUFFER-FIELD("Bezeichnung") NO-ERROR.
IF VALID-HANDLE (hd) THEN DO:

  // Suchen und Setzen der Artikelbezeichnung
  FIND FIRST bS_ArtikelSpr
    WHERE bS_ArtikelSpr.Firma   = hbDB::Firma
      AND bS_ArtikelSpr.Artikel = hbDB::Artikel
      AND bS_ArtikelSpr.Sprache = "D"
      NO-LOCK.


   IF AVAILABLE bS_ArtikelSpr THEN
     ci = bS_ArtikelSpr.Bezeichnung[1] + "~n"
        + bS_ArtikelSpr.Bezeichnung[2] + "~n"
        + bS_ArtikelSpr.Bezeichnung[3] + "~n"
        + bS_ArtikelSpr.Bezeichnung[4] NO-ERROR.
    ELSE
      ci = "X".
  hd:BUFFER-VALUE = ci.
END.
```

3.1.11.2 Die Logik für das Feld EndproduktBezeichnung

In der Tabelle S_ArtikelSpr wird nach einem Datensatz gesucht, der dem aktuellen Artikel, der Sprache Deutsch und der Firma entspricht. Wird ein passender Datensatz gefunden, so wird die Artikelnummer (aus hbDB::Artikel, das heißt das Attribut Artikel aus dem Datensatz der Maske) mit einem Schrägstrich ("/") und der ersten Beschreibung (Bezeichnung[1]) aus dem gefundenen Datensatz kombiniert. Diese Kombination wird dann im Feld "EndproduktBezeichnung" der temporären Tabelle gespeichert. Wird kein passender Datensatz gefunden oder tritt ein Fehler auf, wird stattdessen der Standardwert "X" verwendet.

```
// Verarbeitung des Feldes "EndproduktBezeichnung"
hd = hbTT:BUFFER-FIELD("EndproduktBezeichnung") NO-ERROR.
IF VALID-HANDLE(hd) THEN DO:

  // Kombinieren von Artikelnummer und Bezeichnung
  FIND FIRST bS_ArtikelSpr
    WHERE bS_ArtikelSpr.Firma   = hbDB::Firma
      AND bS_ArtikelSpr.Artikel = hbDB::Artikel
      AND bS_ArtikelSpr.Sprache = "D"
      NO-LOCK.

  IF AVAILABLE bS_ArtikelSpr THEN
    ci = hbDB::Artikel + " / " + bS_ArtikelSpr.Bezeichnung[1] NO-ERROR.
  ELSE
    ci = "X".
  hd:BUFFER-VALUE = ci.
END.
```

3.1.11.3 Die Logik für das Feld MengeUndEinheit

Der Integer-Wert aus dem Feld "Sollmenge" der Business Entity Datenbanktabelle (hbDB::Sollmenge) wird in einen String konvertiert und mit der Einheit "Stk" (für Stück) konkateniert.

```
// Setzen von "MengeUndEinheit"
hd = hbTT:BUFFER-FIELD("MengeUndEinheit") NO-ERROR.
IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = STRING(hbDB::Sollmenge) + " Stk".
```

3.1.11.4 Die Logik für das Feld GeoeffnetAm

Zuerst wird geprüft, ob das Feld "GeoeffnetAm" in der temporären Tabelle (hbTT) vorhanden ist. Wenn das Feld existiert (VALID-HANDLE(hd) ist wahr), wird sein Wert auf das aktuelle Datum (TODAY) gesetzt. Diese Operation weist dem Feld "GeoeffnetAm" das aktuelle Datum zu, das typischerweise verwendet wird, um den Zeitpunkt der Erstellung oder Öffnung eines Datensatzes zu dokumentieren.

```
hd = hbTT:BUFFER-FIELD("GeoeffnetAm") NO-ERROR.
IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = TODAY.
```

3.1.11.5 Die Logik für das Feld PrioritaetUndBez

Zuerst wird geprüft, ob das Feld "PrioritaetUndBez" in der temporären Tabelle vorhanden ist. Wenn ja, wird in der Tabelle M_PrioritaetSpr nach einem Datensatz gesucht, der zur aktuellen Firma, der angegebenen Priorität und der deutschen Sprache passt. Wird ein solcher Datensatz gefunden, so wird ein String erzeugt, der den numerischen Wert der Priorität (aus hbDB::Prioritaet) mit einem Schrägstrich ("/") und der zugehörigen Bezeichnung aus M_PrioritaetSpr kombiniert. Dieser kombinierte String wird dann im Feld "PrioritaetUndBez" der temporären Tabelle gespeichert. Wird kein passender Datensatz gefunden oder tritt ein Fehler auf, wird stattdessen der Standardwert "X" verwendet.

```
// Verarbeitung des Feldes "PrioritaetUndBez"
hd = hbTT:BUFFER-FIELD("PrioritaetUndBez") NO-ERROR.
IF VALID-HANDLE (hd) THEN DO:

  // Suchen und Kombinieren von Priorität und Bezeichnung
  FIND FIRST bM_PrioritaetSpr
    WHERE bM_PrioritaetSpr.Firma       = hbDB::Firma
      AND bM_PrioritaetSpr.Prioritaet  = hbDB::Prioritaet
      AND bM_PrioritaetSpr.Sprache     = "D"
      NO-LOCK.

   IF AVAILABLE bM_PrioritaetSpr THEN
     ci = STRING(hbDB::Prioritaet) + " / " + bM_PrioritaetSpr.Bezeichnung NO-ERROR.
    ELSE
      ci = "X".
  hd:BUFFER-VALUE = ci.
END.
```

3.1.11.6 Die Logik für das Feld LagerOrtUndBez

Zuerst wird geprüft, ob das Feld "LagerOrtUndBez" in der temporären Tabelle vorhanden ist. Wenn dies der Fall ist, wird in der Tabelle ML_OrtSpr nach einem Datensatz gesucht, der zur aktuellen Firma, zum Lagerort und zur Sprache Deutsch passt. Wenn ein solcher Datensatz gefunden wird, wird eine Zeichenkette erzeugt, die den Wert des Lagerorts (aus bML_OrtSpr.LagerOrt) mit einem Schrägstrich ("/") und der entsprechenden Bezeichnung aus ML_OrtSpr kombiniert. Dieser kombinierte String wird dann im Feld "LagerOrtUndBez" der temporären Tabelle gespeichert. Wird kein passender Datensatz gefunden oder tritt ein Fehler auf, wird stattdessen der Standardwert "X" verwendet.

```
// Verarbeitung des Feldes "LagerOrtUndBez"
hd = hbTT:BUFFER-FIELD("LagerOrtUndBez") NO-ERROR.
IF VALID-HANDLE (hd) THEN DO:

  FIND FIRST bML_OrtSpr
    WHERE bML_OrtSpr.Firma       = hbDB::Firma
      AND bML_OrtSpr.LagerOrt    = hbDB::LagerOrt
      AND bML_OrtSpr.Sprache     = "D"
      NO-LOCK.

   IF AVAILABLE bML_OrtSpr THEN
     ci = STRING(bML_OrtSpr.LagerOrt) + " / " + bML_OrtSpr.Bezeichnung NO-ERROR.
    ELSE
      ci = "X".
  hd:BUFFER-VALUE = ci.
END.
```

3.1.12 Calculated Fields be-PP_StkZeile

für die be-PP_StkZeile wurden insgesamt die Felder "BarcodeCode", "BarcodeMaterialBild", "KurzBezeichnungMenge", "MaterialBedarfsMenge" und "BezeichnungArtikelSpr" hinzugefügt.

3.1.12.1 Die Logik für das Feld BarcodeCode

Zuerst wird geprüft, ob das Feld "BarcodeCode" in der temporären Tabelle vorhanden ist. Ist dies der Fall, wird der Wert dieses Feldes aus drei Elementen zusammengesetzt: dem Präfix "000", der Rückmeldungsnummer (hbDB::RueckMeldeNr) und hbDB::StructCode. Dieser zusammengesetzte String wird dann im Feld "BarcodeCode" der temporären Tabelle gespeichert, wodurch ein eindeutiger Barcodecode erzeugt wird.

```
    // Setzen des Barcode-Codes
    hd = hbTT:BUFFER-FIELD("BarcodeCode") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "000" + hbDB::RueckMeldeNr + hbDB::StructCode.
```

3.1.12.2 Die Logik für das Feld BarcodeMaterialBild

Zuerst wird geprüft, ob das Feld "BarcodeMaterialBild" in der temporären Tabelle vorhanden ist. Ist dies der Fall, wird ein Prozess zur Erzeugung und Speicherung eines Barcodestrings gestartet. Zunächst wird der Barcodewert erzeugt, indem das Präfix "000" mit der Rückmeldenummer (hbDB::RueckMeldeNr) und hbDB::StructCode kombiniert wird. Anschließend wird eine URL für den Online-Barcode-Generator (TEC-IT) konkatiniert, die spezifische Parameter für Barcode-Typ, -Größe und -Format enthält.

Es wird über den generierten Pfad geprüft, ob das Barcodebild bereits lokal vorhanden ist. Falls nicht, wird ein PowerShell-Befehl generiert und ausgeführt, der das Barcode-Bild von der URL herunterlädt und lokal speichert. Schließlich wird der Dateipfad des Barcodebildes im Feld "BarcodeMaterialBild" der temporären Tabelle gespeichert.

Dieser Prozess stellt sicher, dass für jeden eindeutigen Barcodewert ein entsprechendes Bild verfügbar ist, entweder durch Neuerstellung oder durch Verwendung eines bereits vorhandenen Bildes.

```
    // Generieren und Speichern des Barcode-Bildes über einen Shell Command
    hd = hbTT:BUFFER-FIELD("BarcodeMaterialBild") NO-ERROR.
    IF VALID-HANDLE(hd) THEN
    DO:
      DEFINE VARIABLE cUri          AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cFilePath     AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cCommand      AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cbarcodeValue AS CHARACTER NO-UNDO.
      DEFINE VARIABLE lFileExists   AS LOGICAL NO-UNDO.

      // Erstellen des Barcode-Wertes und der URL für die Barcode-Generierung
      cbarcodeValue = "000" + hbDB::RueckMeldeNr + hbDB::StructCode.
      cUri          = "https://barcode.tec-it.com/barcode.ashx?data=" + cbarcodeValue +
"`&code=Code128`&translate-
esc=off`&dpi=94`&modulewidth=0.19`&height=6`&unit=Fit`&imagetype=png`&download=true`&hidehrt=t
rue".
      cFilePath     = SUBSTITUTE("e:/proalpha/pa-de-9/demo/zv/pic/" + "&1", cBarcodeValue +
"Barcode" + ".png").

      // Überprüfen, ob die Datei bereits existiert
      lFileExists   = SEARCH(cFilePath) <> ?.

      // Wenn die Datei nicht existiert, herunterladen und speichern
      IF NOT lFileExists THEN
      DO:
        cCommand = SUBSTITUTE('powershell -Command Invoke-WebRequest -Uri "&1" -OutFile "&2"',
cUri, cFilePath).
        OS-COMMAND SILENT VALUE(cCommand).
      END.

      hd:BUFFER-VALUE = cFilePath.
    END.
```

3.1.12.3 Die Logik für das Feld KurzBezeichnungMenge

Zuerst wird geprüft, ob das Feld "KurzBezeichnungMenge" in der temporären Tabelle vorhanden ist. Wenn ja, wird in der Tabelle S_MengenEinheitSpr nach einem Datensatz gesucht, der zur angegebenen Mengeneinheit (hbDB::MengenEinheit) und zur Sprache Deutsch passt. Wird ein solcher Datensatz gefunden, so wird die Kurzbezeichnung (KurzBez) aus diesem Datensatz extrahiert und im Feld "KurzBezMenge" der temporären Tabelle gespeichert. Wird kein passender Datensatz gefunden, wird stattdessen der Standardwert "X" verwendet.

```
    // Setzen der Kurzbezeichnung für die Mengeneinheit
```

```
hd = hbTT:BUFFER-FIELD("KurzBezeichnungMenge") NO-ERROR.
IF VALID-HANDLE(hd) THEN DO:
  FIND FIRST bS_MengenEinheitSpr
    WHERE bS_MengenEinheitSpr.Firma           = ""
      AND bS_MengenEinheitSpr.MengenEinheit   = hbDB::MengenEinheit
      AND bS_MengenEinheitSpr.Sprache         = "D"
      NO-LOCK.

   IF AVAILABLE bS_MengenEinheitSpr THEN
       ci = bS_MengenEinheitSpr.KurzBez.
   ELSE
      ci = "X".
   hd:BUFFER-VALUE = ci.
END.
```

3.1.12.4 Die Logik für das Feld MaterialBedarfsMenge

Zuerst wird geprüft, ob das Feld "MaterialBedarfsMenge" in der temporären Tabelle vorhanden ist. Dann wird die Mengeneinheit des Datensatzes geprüft. Wenn die Mengeneinheit 11 oder 10 ist (was für spezifische Einheiten wie Stück oder Meter stehen kann), wird der Wert des Feldes "Bedarfsmenge" aus der Datenbanktabelle mit der Zeichenkette ".000" konkateniert. Dadurch werden drei Dezimalstellen hinzugefügt. Für alle anderen Mengeneinheiten wird der Wert des Feldes "Bedarfsmenge" unverändert übernommen. Der resultierende Wert wird dann im Feld "MaterialBedarfsMenge" der temporären Tabelle gespeichert.

```
// Setzen der Materialbedarfsmenge
hd = hbTT:BUFFER-FIELD("MaterialBedarfsMenge") NO-ERROR.
IF hbDB::Mengeneinheit = 11 OR hbDB::Mengeneinheit = 10 THEN DO:
  IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = hbDB::Bedarfsmenge + ",000".
END.
ELSE
IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = hbDB::Bedarfsmenge.
```

3.1.12.5 Die Logik für das Feld BezeichnungArtikelSpr

Zuerst wird geprüft, ob das Feld "BezeichnungArtikelSpr" in der temporären Tabelle existiert. Wenn ja, wird in der Tabelle S_ArtikelSpr nach einem Datensatz gesucht, der zur aktuellen Firma, zum spezifischen Artikel und zur Sprache Deutsch passt. Wenn ein solcher Datensatz gefunden wird, wird der Wert des ersten Bezeichnungsfeldes (Bezeichnung[1]) aus dem Array extrahiert. Dieser Wert wird dann im Feld "BezeichnungArtikelSpr" der temporären Tabelle gespeichert. Wird kein passender Datensatz gefunden oder tritt ein Fehler auf, wird stattdessen der Standardwert "X" verwendet. Diese Logik ermöglicht es, die korrekte Artikelbezeichnung in der gewünschten Sprache für einen bestimmten Artikel zu setzen.

```
hd = hbTT:BUFFER-FIELD("BezeichnungArtikelSpr") NO-ERROR.
IF VALID-HANDLE(hd) THEN DO:

  FIND FIRST bS_ArtikelSpr
    WHERE bS_ArtikelSpr.Firma   = hbDB::Firma
      AND bS_ArtikelSpr.Artikel = hbDB::Artikel
      AND bS_ArtikelSpr.Sprache = "D"
      NO-LOCK NO-ERROR.

   IF AVAILABLE bS_ArtikelSpr THEN
     ci = bS_ArtikelSpr.Bezeichnung[1].
    ELSE
      ci = "X".
  hd:BUFFER-VALUE = ci.
END.
```

3.1.13 Calculated Fields be-MB_Aktivitaet

Für die be-PP_StkZeile wurden insgesamt die Felder "SollZeitGesamt", "MinZeitEinheit", "BarcodeArbeitsFolgen", "BarcodeFolgenBild" und "RueckmeldungText" hinzugefügt.

3.1.13.1 Die Logik für das Feld SollZeitGesamt

Zuerst wird geprüft, ob das Feld "SollZeitGesamt" in der temporären Tabelle existiert. Ist dies der Fall, wird der Wert dieses Feldes berechnet, indem zwei Werte aus der Datenbanktabelle addiert werden: die Sollzeit für Rüsten (hbDB::soll_tr) und die Sollzeit für Fertigung (hbDB::soll_te). Die Summe dieser beiden Zeiten ergibt die gesamte Sollzeit, die dann im Feld "SollZeitGesamt" der temporären Tabelle gespeichert wird.

3.1.13.2 Die Logik für das Feld MinZeitEinheit

Zuerst wird geprüft, ob das Feld "MinZeitEinheit" in der temporären Tabelle existiert. Wenn ja, wird der Wert dieses Feldes auf den String "Min" gesetzt.

```
    // Setzen der Zeiteinheit auf "Min"
    hd = hbTT:BUFFER-FIELD("MinZeitEinheit") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "Min".
```

3.1.13.3 Die Logik für das Feld BarcodeArbeitsFolgen

Zuerst wird geprüft, ob das Feld "BarcodeArbeitsFolgen" in der temporären Tabelle existiert. Ist dies der Fall, so wird der Wert dieses Feldes durch Verkettung von vier Elementen gebildet: dem Präfix "000", dem Teilprozess (hbDB::Teilprozess), der Aktivitätsart (hbDB::AktArt) und der Aktivitätsposition (hbDB::AktPos). Dieser zusammengesetzte String wird dann im Feld "BarcodeArbeitsFolgen" der temporären Tabelle gespeichert. Durch diese Kombination wird für jede Arbeitsfolge ein eindeutiger Barcode erzeugt, der Informationen über den Teilprozess, die Art der Aktivität und ihre Position im Gesamtprozess enthält.

```
    // Generieren des Barcodes für Arbeitsfolgen
    hd = hbTT:BUFFER-FIELD("BarcodeArbeitsFolgen") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "000" + hbDB::Teilprozess + hbDB::AktArt +
hbDB::AktPos.
```

3.1.13.4 Die Logik für das Feld BarcodeFolgenBild

Zuerst wird geprüft, ob das Feld "BarcodeFolgenBild" in der temporären Tabelle vorhanden ist. Wenn dies der Fall ist, wird ein Prozess gestartet, um ein Barcode-Bild für Arbeitsfolgen zu erzeugen und zu speichern. Aus dem Präfix "000", dem Teilprozess, dem Aktivitätstyp und der Aktivitätsposition wird ein Barcodewert erzeugt. Es wird eine URL erzeugt, die auf einen Online-Barcode-Generator (TEC-IT) mit spezifischen Parametern für Barcode-Typ, -Größe und -Format verweist.

Ein String für das zu speichernde Barcodebild wird auf der Grundlage des Barcodewertes erstellt. Es wird geprüft, ob das Barcode-Bild bereits lokal vorhanden ist. Falls nicht, wird ein PowerShell-Befehl generiert und ausgeführt, der das Barcode Bild von der URL herunterlädt und lokal speichert. Schließlich wird der Dateipfad des Barcodebildes im Feld "BarcodeFolgenBild" der temporären Tabelle gespeichert.

```
    // Generieren und Speichern des Barcode-Bildes für Arbeitsfolgen
    hd = hbTT:BUFFER-FIELD("BarcodeFolgenBild") NO-ERROR.
    IF VALID-HANDLE(hd) THEN
    DO:
      DEFINE VARIABLE cUri          AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cFilePath     AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cCommand      AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cbarcodeValue AS CHARACTER NO-UNDO.
      DEFINE VARIABLE lFileExists   AS LOGICAL NO-UNDO.

      // Erstellen des Barcode-Wertes und der URL für die Barcode-Generierung
      cbarcodeValue = "000" + hbDB::Teilprozess + hbDB::AktArt + hbDB::AktPos.
      cUri          = "https://barcode.tec-it.com/barcode.ashx?data=" + cbarcodeValue +
"`&code=Code128`&translate-
esc=off`&dpi=94`&modulewidth=0.19`&height=6`&unit=Fit`&imagetype=png`&download=true`&hidehrt=t
rue".
      cFilePath     = SUBSTITUTE("e:/proalpha/pa-de-9/demo/zv/pic/" + "&1", cBarcodeValue +
"Barcode" + ".png").

      // Überprüfen, ob die Datei bereits existiert
      lFileExists   = SEARCH(cFilePath) <> ?.

      // Wenn die Datei nicht existiert, herunterladen und speichern
      IF NOT lFileExists THEN
      DO:
        cCommand = SUBSTITUTE('powershell -Command Invoke-WebRequest -Uri "&1" -OutFile "&2"',
cUri, cFilePath).
        OS-COMMAND SILENT VALUE(cCommand).
      END.

      hd:BUFFER-VALUE = cFilePath.
    END.
```

3.1.13.5 Die Logik für das Feld RueckmeldungText

Zuerst wird geprüft, ob das Feld "RueckmeldungText" in der temporären Tabelle existiert. Wenn ja, wird der Wert dieses Feldes abhängig vom Status der Rückmeldung in der Datenbanktabelle gesetzt. Dazu wird eine bedingte Zuweisung verwendet: Wenn das Feld "Rückmeldung" in der Datenbanktabelle wahr (TRUE) ist, wird der Text "Rückmeldung erforderlich" gesetzt. Andernfalls, wenn es falsch (FALSE) ist, wird der Text "Rückmeldung nicht erforderlich" gesetzt.

```
    // Setzen des Rückmeldungstextes basierend auf dem Rückmeldungsstatus
    hd = hbTT:BUFFER-FIELD("RueckmeldungText") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = IF hbDB::Rueckmeldung THEN "Rückmeldung
erforderlich" ELSE "Rückmeldung nicht erforderlich".
```

3.1.14 Calculated Field be-P_ZeichnungArtikel

Für die be-P_ZeichnungArtikel wurde insgesamt das Feld "CadNumberPath hinzugefügt.

3.1.14.1 Die Logik für das Feld CadNumberPath

Zuerst wird geprüft, ob das Feld "CadNumberPath" in der temporären Tabelle existiert. Wenn ja, wird in der Tabelle PMM_Drawing nach einem Datensatz gesucht, der zur aktuellen Firma und zur spezifischen Zeichnungsnummer (hbDB::Zeichnung) passt. Wird ein solcher Datensatz gefunden, wird der Pfad zur technischen Zeichnung erstellt. Dabei wird die CAD-Nummer aus dem gefundenen Datensatz verwendet, wobei die Dateiendung ".dwg" durch ".png" ersetzt wird.

Wird kein passender Datensatz gefunden, wird stattdessen der Standardwert "X" verwendet. Der konstruierte Pfad oder der Standardwert wird dann im Feld "CadNumberPath" der temporären Tabelle gespeichert.

```
    hd = hbTT:BUFFER-FIELD("CadNumberPath") NO-ERROR.
    IF VALID-HANDLE(hd) THEN DO:

      // Suchen der zugehörigen Zeichnung
      FIND FIRST bPMM_Drawing
          WHERE bPMM_Drawing.Company        = hbDB::Firma
            AND bPMM_Drawing.PMM_Drawing_ID = hbDB::Zeichnung
          NO-LOCK.

          // Setzen der CAD-Nummer (Pfad zur Technischen Zeichnung) oder "X" wenn nicht
verfügbar
          // tauschen der .dwg Endung im String mit .png "Vorläufig"
          IF AVAILABLE bPMM_Drawing THEN
           ci = "E:/proalpha/pa-de-9/demo/zv/cad/" + REPLACE(bPMM_Drawing.CadNumber, ".dwg",
".png").
           ELSE
            ci = "X".
       hd:BUFFER-VALUE = ci.
    END.
```

Der gesamte Quellcode der Datenbankmodelle befindet sich im Anhang unter D. Datenbankmodell.

3.2 Eingabemaskenauswahl in der graphischen Oberfläche

Zur Weiterleitung an die Zeichnungsmaske und die Prozessdatenblattmaske aus der Laufkarte heraus wurden Buttons in die Laufkartenmaske geplant. Bei dem Design der Laufkarte liegt mittig rechts Platz für diese zwei Buttons. Die Weiterleitung zu den verschiedenen Masken in OF-1 wird durch eine einfache und intuitive Methode realisiert. Dazu wird die von OF-1 zur Verfügung gestellte Eigenschaft "Screen" des Button-Objekts verwendet. Diese Eigenschaft ermöglicht die direkte Auswahl des gewünschten Zielbildschirms, der sich öffnen soll, wenn der Button betätigt wird. Um diese Funktionalität zu nutzen, muss der Entwickler lediglich die "Screen"-Eigenschaft in den Eigenschaften des Button-Objekts auswählen. Dort wird eine Liste aller im Projekt verfügbaren Bildschirme angezeigt. Der Entwickler wählt dann die Maske aus, zu der der Button weiterleiten soll.

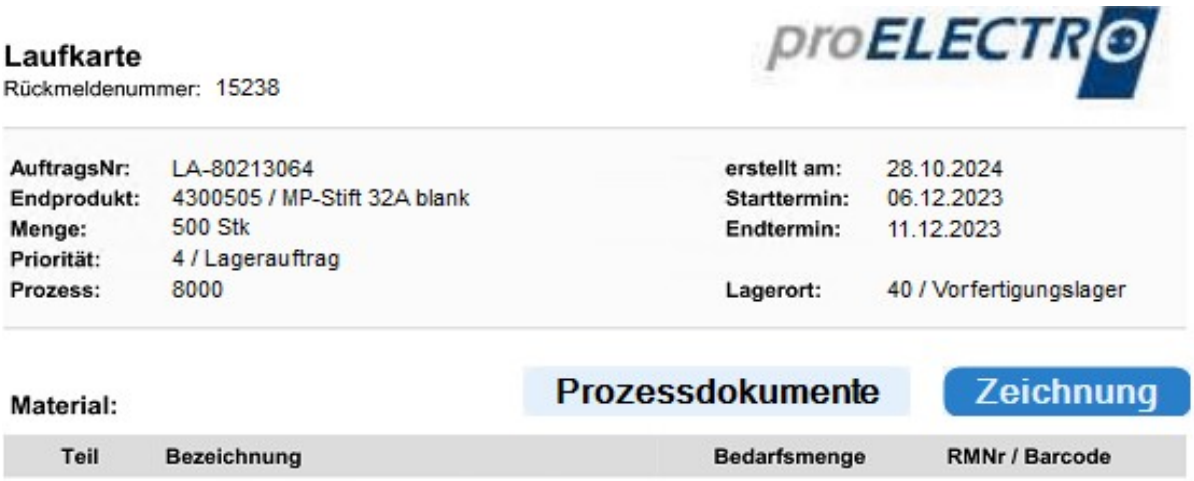


Abbildung 3: Laufkarte mit Buttons

4 Qualitätssicherung

Die Funktionalität eines neu implementierten Features wurde unmittelbar im Rahmen der Implementierung getestet.

4.1 Code Review

Die Code-Reviews für dieses Projekt wurden von einem erfahrenen Entwickler der IAP GmbH durchgeführt. Der Senior Entwickler verfügt über umfassende Kenntnisse in OpenEdge- und OF-1 Programmierung und ist zudem aktiv am Entwicklungsprozess von OF-1 und proAlpha beteiligt.

Im Rahmen der Code-Reviews hat er den erstellten Programmcode und die erstellten Business-Entity sorgfältig geprüft und analysiert. Dabei lag sein Augenmerk besonders auf der Identifizierung potenzieller Fehler und Optimierungsmöglichkeiten. Ein konkretes Beispiel hierfür war die ursprüngliche Struktur der Business Entity für die be_P_Zeichnungsartikel, diese wurde in die be_PP-Auftrag integriert. Bei der Überprüfung der Log-Datei wurde festgestellt, dass eine notwendige For-Each-Abfrage des Custom-Browse Objektes für die P_Zeichnungsartikel fehlte.

Um dieses Problem zu lösen, wurde beschlossen, eine neue Business-Entity speziell für die Zeichnungsartikel zu erstellen. Diese Entscheidung war notwendig, da sich die relevanten Zeichnungsdaten in den Tabellen P_ZeichnungArtikel und PMM_Drawing befinden. Die bestehende Business Entity be_PP-Auftrag wurde bereits für die Ermittlung der Artikeldaten über ein dynamisches Fill-In verwendet und konnte somit nicht für die Abfrage der Zeichnungsartikel verwendet werden.

Durch die Einführung einer eigenen Business-Entity für die Zeichnungsartikel konnte eine klare Trennung der Datenstrukturen erreicht werden. Dadurch wurden Konflikte bei der Datenabfrage vermieden und die Datenorganisation erheblich verbessert. Durch die Expertise des Entwicklers konnte sowohl die Qualität als auch die Performance der Anwendung deutlich gesteigert werden.

4.2 Durchführung der Tests

Die Anzeige der korrekten Eingabemasken mit korrekten Daten und das Speichern dieser, wurde durch reguläre Verwendung des Screens mit mehrfachem Öffnen, Schließen, Verändern der Werte in den Eingabekomponenten getestet. Hierbei ist aufgefallen, dass die technischen Zeichnungen in der Zeichnungsmaske nicht angezeigt wurden.

Nach Rücksprache mit den Experten von OF-1 stellte sich heraus, dass das Problem auf eine Einschränkung des Frameworks zurückzuführen ist. OF-1 unterstützt derzeit keine .dwg-Dateien, die üblicherweise für technische Zeichnungen verwendet werden. Diese Dateien enthalten Vektorgrafiken, für deren Darstellung spezielle Objekte benötigt werden. Da OF-1 diese Objekte nicht zur Verfügung stellt, können die Vektorgrafiken aus den .dwg-Dateien nicht korrekt angezeigt werden.

Diese Einschränkung erklärt, warum die technischen Zeichnungen in der Zeichnungsmaske nicht

sichtbar sind.

4.3 Fehlerkorrektur

Um das Problem der Anzeige von technischen Zeichnungen zu lösen, kann man das DXF-Format verwenden, anstatt direkt mit .dwg-Dateien zu arbeiten. DXF, das für Drawing Exchange Format steht, speichert die Daten als ASCII-Text und ist daher für verschiedene Systeme, einschließlich OpenEdge, leichter lesbar und interpretierbar.

Eine vielversprechende Möglichkeit ist die Verwendung der .NET-Bibliothek ACadSharp. Diese Bibliothek kann sowohl DWG- als auch DXF-Dateien lesen und steht unter der MIT-Lizenz, was bedeutet, dass sie kostenlos und kommerziell genutzt werden kann. Um ACadSharp in OpenEdge zu integrieren, muss die Bibliothek heruntergeladen und über die Datei assemblies.xml in das Projekt eingebunden werden.

Nach dem Hinzufügen der entsprechenden "using"-Anweisung stehen dann alle Funktionen der Bibliothek zur Verfügung. Zusätzlich könnte es sinnvoll sein, einen .NET-Wrapper zu erstellen, der eine speziell auf die geplanten Arbeiten zugeschnittene Schnittstelle bietet. Dieser Wrapper würde die Funktionalität von ACadSharp nutzen und eine vereinfachte Schnittstelle für die spezifischen Anforderungen des Projekts bereitstellen.

Mithilfe dieses Wrappers oder direkt über ACadSharp könnten die relevanten Daten aus den DXF-Dateien extrahiert und in einem für OF-1 geeigneten Format aufbereitet werden. Mit diesem Ansatz wäre es möglich, die technischen Zeichnungen zu lesen und zu verarbeiten, ohne dass OF-1 direkt .dwg-Dateien unterstützen muss. Die Verwendung einer etablierten Bibliothek wie ACadSharp in Kombination mit einem Wrapper bietet Flexibilität und Effizienz bei der Implementierung. Jedoch wurde diese Fehlerkorrektur nicht umgesetzt, da die Projektzeit dafür nicht mehr gereicht hat.

Ich habe mich vorläufig für eine andere Lösung entschieden, um die Entwicklung der Zeichnungsmaske voranzutreiben, ohne durch die derzeitige Einschränkung bei der Anzeige von .dwg-Dateien blockiert zu sein. Zunächst konzentrierte ich mich auf die Vervollständigung der Logik für die Dateipfade der relevanten Zeichnungen. Diese Pfade werden zunächst als Textfelder in der Zeichnungsmaske angezeigt. Dieser Schritt ermöglicht es, die Maske so weit wie möglich zu vervollständigen und funktionsfähig zu machen, auch wenn die eigentlichen Zeichnungen noch nicht angezeigt werden können. Für die Zukunft sehe ich zwei Möglichkeiten: Entweder warte ich darauf, dass in OF-1 die Unterstützung für .dwg-Dateien implementiert wird, oder ich plane die Entwicklung einer alternativen Lösung wie oben beschrieben. Eine weitere Alternative kann die Konvertierung der .dwg-Dateien in ein von OF-1 unterstütztes Format wie .png über eine Schnittstelle sein.

Sobald eine dieser Optionen implementiert ist, kann ich die bestehende Implementierung leicht anpassen. Der Dateipfad, der derzeit als Text angezeigt wird, kann dann direkt in ein Bildobjekt eingebettet werden. Dies würde es ermöglichen, die tatsächlichen Zeichnungen in der Maske anzuzeigen, ohne größere Änderungen an der bereits entwickelten Struktur vornehmen zu müssen. Mein Ansatz zeigt eine flexible und vorausschauende Herangehensweise. Er ermöglicht es, die Entwicklung fortzusetzen und gleichzeitig offen zubleiben für zukünftige Verbesserungen, sobald die technischen Voraussetzungen gegeben sind.
Ein Screenshot der vorläufigen Lösung befindet sich im Anhang unter d. Zeichnung mit Dateipfaden.

5 Projektabschluss

5.1 SOLL-IST Vergleich

Die MUSS-Kriterien wurden erfolgreich umgesetzt, indem Masken für die Laufkarte und die zugehörigen Zeichnungen erstellt wurden, die in der festgelegten Reihenfolge angezeigt werden und eine mandantenabhängige Steuerung der Daten in der Zeichnungsmaske ermöglichen, während die Entscheidbarkeit auf Artikelebene für die Anzeige von Folgedokumenten über ein Eingabefeld sicherstellt ist.

Die Kann-Kriterien wurden ebenfalls berücksichtigt, indem jedes Dokument separat über eine eigene

Oberflächenmaske zugeordnet wird und die Befüllung der relevanten Datenfelder automatisch über die Datenbankanbindung und entsprechende Methoden des Backends erfolgt.

Allerdings konnte das Kriterium für das Prozessdatenblatt noch nicht vollständig erfüllt werden. Dies stellt jedoch keinen gravierenden Mangel dar, da die bereits implementierten Funktionen eine solide Basis bieten, auf der das fehlende Element in zukünftigen Entwicklungen aufgebaut werden kann. Die Integration der Prozessdatenblattmaske wird daher als nächste Priorität angesehen, um das Add-on weiter zu optimieren und zu vervollständigen.

5.2 SOLL-IST Zeitvergleich

In den einzelnen Projektphasen gab es Abweichungen der geplanten und genutzten Zeit, jedoch wurde die gesamte geplante Zeit eingehalten. Insgesamt wurden 4 Stunden anders als geplant verwendet.

Projektphase	Geplante Zeit (Stunden)	Genutzte Zeit (Stunden)	Abweichung (Stunden)
Entwicklung der digitalen Laufzettel-Oberfläche	5	10	5
Entwicklung der digitalen Zeichnungs-Oberfläche	3	6	3
Entwicklung der digitalen Prozessdatenblatt-Oberfläche	5	1	-4
Integration der Echtzeit-Datenbank	6	8	2
Entwicklung der Backend-Logik	13	15	2
Entwicklung des Add-ons	**8**	**4**	**-4**

Tabelle 5: SOLL-IST Zeitvergleich

Durch die vorherige Unterschätzung der unbekannten proAlpha Datenbankstruktur und Umgebung wurde zusätzliche Zeit verwendet, nach den Datenbankverbindungen zu suchen und die neue proAlpha Umgebung über die lokal installierte virtuelle Maschine kennenzulernen. Zudem wurde durch die ungewöhnliche Relationsstruktur zu den Laufkarten mit den dazugehörigen Dokumenten mehr Zeit bei der Erstellung und Datenbankanbindung der Eingabemasken verwendet.

Die danach gewählte Lösung für eine automatische Einsortierung von Laufkarten ließ sich dann aber schneller implementieren als erwartet, wodurch der Rückstand eingeholt werden konnte. Aufgrund der unvollständigen Ausführung des Projekts wurde auf eine Einspielung in die proAlpha Live-Umgebung verzichtet. Die dadurch gewonnene Zeit wurde in die Dokumentation investiert.

5.3 Übergabe

Nach dem Review mit dem Ausbildenden wurden die Änderungen in das interne SVN eingecheckt. Für das Deployment müssen die Screens und der Code für diese über ein von dem Nutzer angefragtes Update an das SVN, an alle Nutzer verteilt werden.

5.4 Fazit

Durch dieses Projekt wurde mir bewusst, dass ich meine eigenen Fähigkeiten überschätze. Das ursprünglich geplante Volumen an Features konnte ich nur teilweise und insbesondere in der Automatisierung nicht zufriedenstellend umsetzen.

Im Umgang mit dem OF-1 Framework konnte ich auch neue Erfahrungen sammeln. Die Erstellung von Screens ist im Rahmen der Umschulung regelmäßig vorgekommen, aber die eigene Implementierung zum Synchronisieren der Daten zwischen Client und Server war für mich eine Neuerung. Die komplett eigenständige Durchführung eines Projekts hat mich auch meine mangelnde Erfahrung in den wirtschaftlichen und planenden Tätigkeiten erkennen lassen.

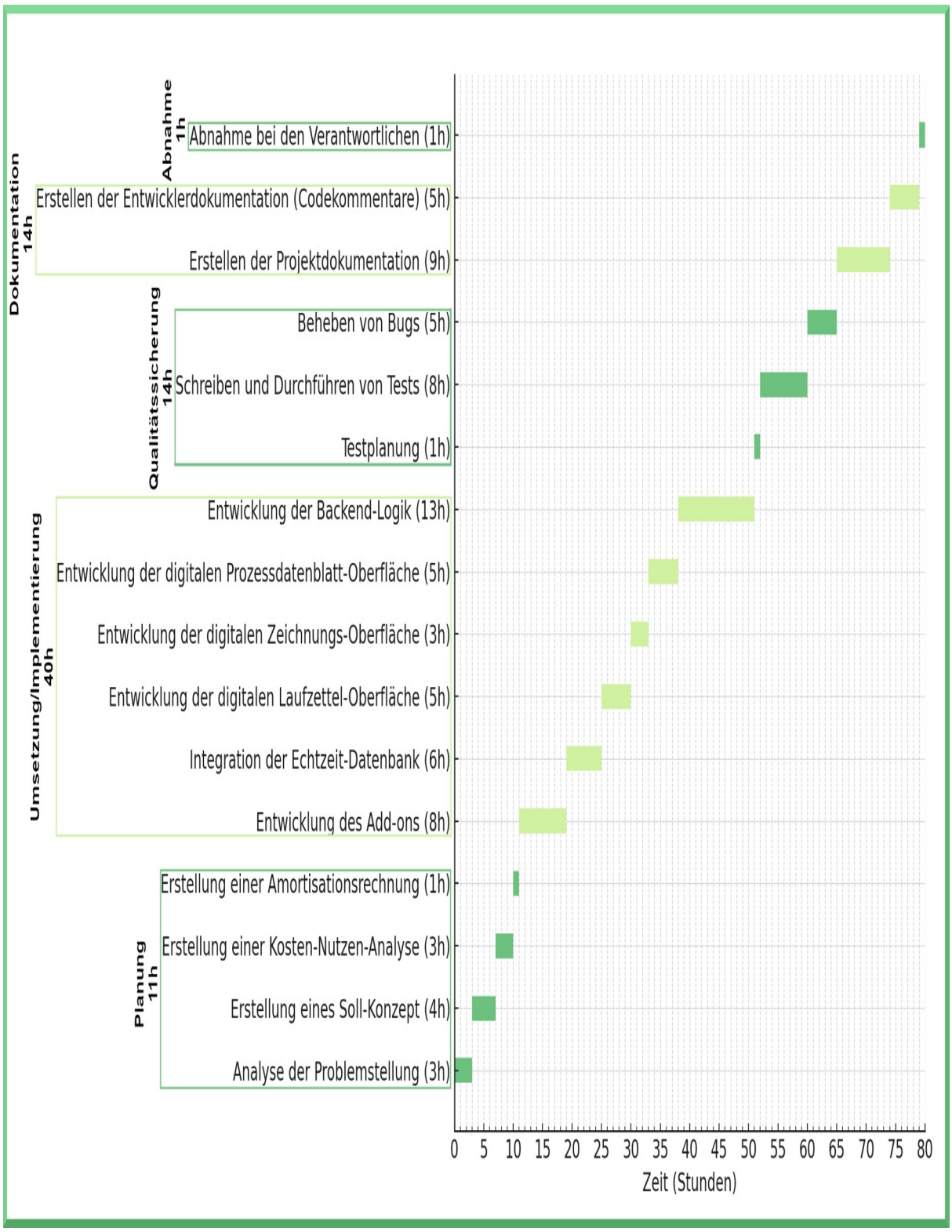
Insgesamt war das Projekt sehr interessant zu bearbeiten und ich freue mich darauf im Anschluss an dieses Projekts die fehlenden Aspekte implementieren zu dürfen.

ANHANG

A. Tabellenverzeichnis

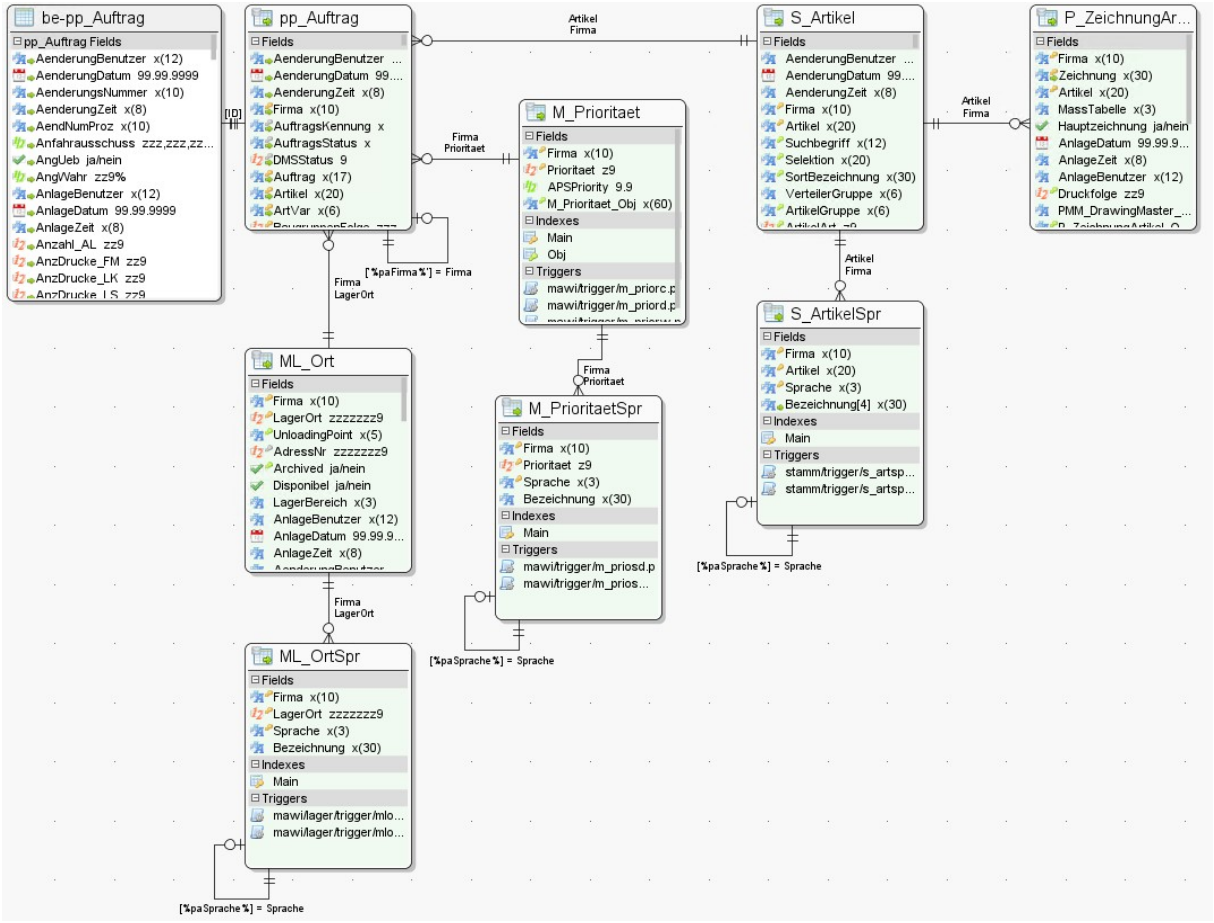
B. Abbildungsverzeichnis

C. Gantt-Diagramm



D. Datenbankmodell

a. be-PP_Auftrag

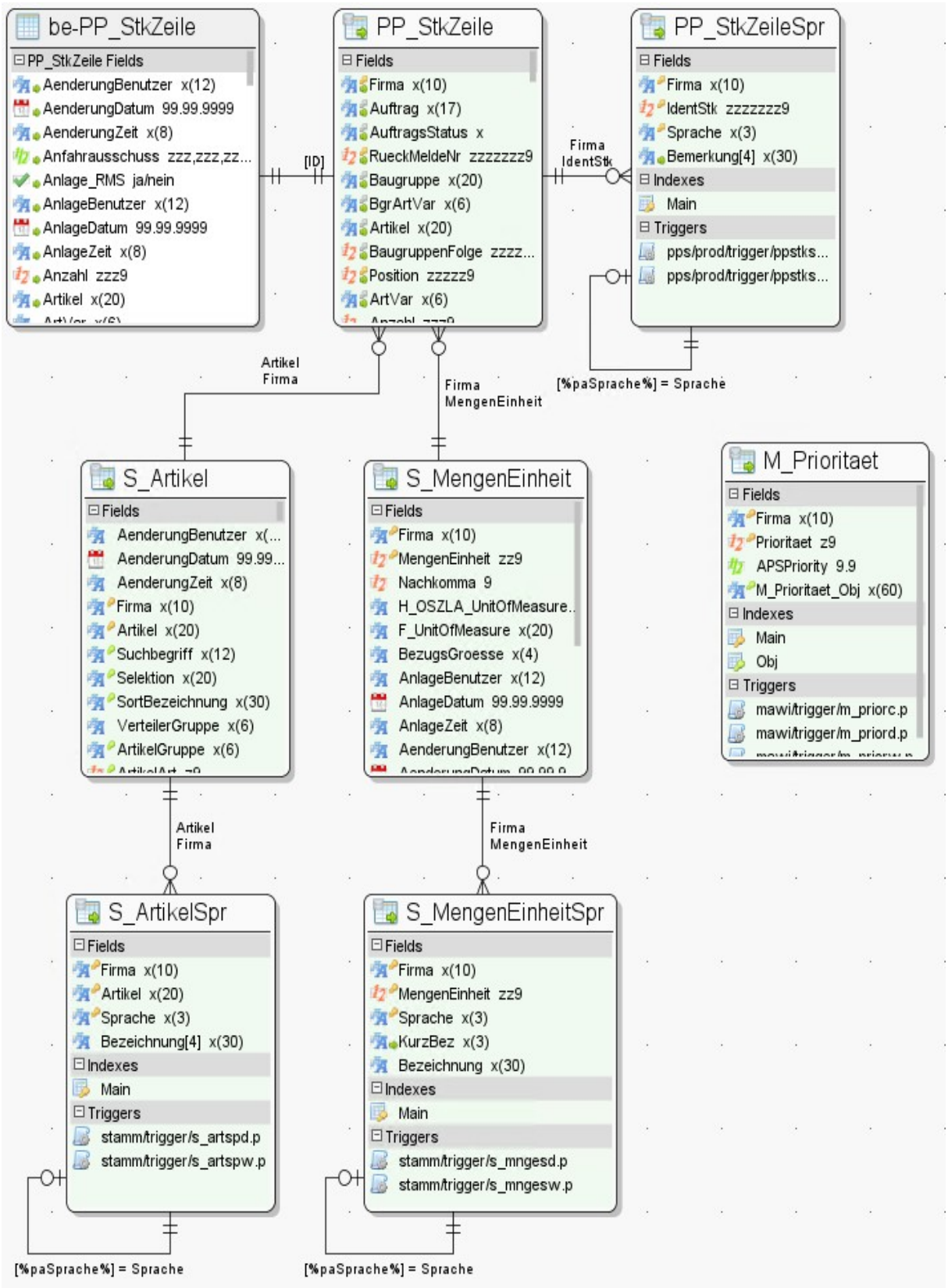


b. Backend Logik aus der be-pp_auftrag.cls

```
// Importieren benötigter Klassen und Entitäten
USING of1.entities.be-of1.
USING proalpha.entities.be-pp_auftrag-g.
USING of1.cclasses.c-session FROM PROPATH.
USING proalpha.entities.be-pp_auftrag FROM PROPATH.

// Definition der Klasse, die von be-pp_auftrag-g erbt
CLASS proalpha.entities.be-pp_auftrag INHERITS be-pp_auftrag-g :

  // Überschreiben der Methode zum Füllen berechneter Felder
  METHOD OVERRIDE VOID mFillCalculatedFields (INPUT hbVar AS HANDLE):
    DEFINE VARIABLE hbTT AS HANDLE NO-UNDO.
    DEFINE VARIABLE hbDB AS HANDLE NO-UNDO.
    DEFINE VARIABLE hd   AS HANDLE NO-UNDO.
    DEFINE VARIABLE ci   AS CHARACTER NO-UNDO.

    // Definition von Puffern für Datenbanktabellen
    DEFINE BUFFER bS_ArtikelSpr       FOR S_ArtikelSpr.
    DEFINE BUFFER bM_PrioritaetSpr    FOR M_PrioritaetSpr.
    DEFINE BUFFER bML_OrtSpr          FOR ML_OrtSpr.

    // Zuweisen von Handles für temporäre und Datenbank-Tabellen
    hbTT = hbVar::hbTT.
    hbDB = hbVar::hbDB.

    // Verarbeitung des Feldes "Bezeichnung"
    hd = hbTT:BUFFER-FIELD("Bezeichnung") NO-ERROR.
    IF VALID-HANDLE (hd) THEN DO:

      // Suchen und Setzen der Artikelbezeichnung
      FIND FIRST bS_ArtikelSpr
        WHERE bS_ArtikelSpr.Firma    = hbDB::Firma
```

```
          AND bS_ArtikelSpr.Artikel = hbDB::Artikel
          AND bS_ArtikelSpr.Sprache = "D"
          NO-LOCK.


       IF AVAILABLE bS_ArtikelSpr THEN
         ci = bS_ArtikelSpr.Bezeichnung[1] + "~n"
            + bS_ArtikelSpr.Bezeichnung[2] + "~n"
            + bS_ArtikelSpr.Bezeichnung[3] + "~n"
            + bS_ArtikelSpr.Bezeichnung[4] NO-ERROR.
       ELSE
         ci = "X".
     hd:BUFFER-VALUE = ci.
   END.

   // Verarbeitung des Feldes "EndproduktBezeichnung"
   hd = hbTT:BUFFER-FIELD("EndproduktBezeichnung") NO-ERROR.
   IF VALID-HANDLE(hd) THEN DO:

     // Kombinieren von Artikelnummer und Bezeichnung
     FIND FIRST bS_ArtikelSpr
       WHERE bS_ArtikelSpr.Firma    = hbDB::Firma
         AND bS_ArtikelSpr.Artikel = hbDB::Artikel
         AND bS_ArtikelSpr.Sprache = "D"
         NO-LOCK.

     IF AVAILABLE bS_ArtikelSpr THEN
       ci = hbDB::Artikel + " / " + bS_ArtikelSpr.Bezeichnung[1] NO-ERROR.
     ELSE
       ci = "X".
     hd:BUFFER-VALUE = ci.
   END.

   // Setzen von "MengeUndEinheit"
   hd = hbTT:BUFFER-FIELD("MengeUndEinheit") NO-ERROR.
   IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = STRING(hbDB::Sollmenge) + " Stk".

   hd = hbTT:BUFFER-FIELD("GeoeffnetAm") NO-ERROR.
   IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = DATE(TODAY).

   // Verarbeitung des Feldes "PrioritaetUndBez"
   hd = hbTT:BUFFER-FIELD("PrioritaetUndBez") NO-ERROR.
   IF VALID-HANDLE (hd) THEN DO:

     // Suchen und Kombinieren von Priorität und Bezeichnung
     FIND FIRST bM_PrioritaetSpr
       WHERE bM_PrioritaetSpr.Firma       = hbDB::Firma
         AND bM_PrioritaetSpr.Prioritaet  = hbDB::Prioritaet
         AND bM_PrioritaetSpr.Sprache     = "D"
         NO-LOCK.

      IF AVAILABLE bM_PrioritaetSpr THEN
        ci = STRING(hbDB::Prioritaet) + " / " + bM_PrioritaetSpr.Bezeichnung NO-ERROR.
       ELSE
         ci = "X".
     hd:BUFFER-VALUE = ci.
   END.

   // Verarbeitung des Feldes "LagerOrtUndBez"
   hd = hbTT:BUFFER-FIELD("LagerOrtUndBez") NO-ERROR.
   IF VALID-HANDLE (hd) THEN DO:

     FIND FIRST bML_OrtSpr
       WHERE bML_OrtSpr.Firma       = hbDB::Firma
         AND bML_OrtSpr.LagerOrt    = hbDB::LagerOrt
         AND bML_OrtSpr.Sprache     = "D"
         NO-LOCK.

      IF AVAILABLE bML_OrtSpr THEN
        ci = STRING(bML_OrtSpr.LagerOrt) + " / " + bML_OrtSpr.Bezeichnung NO-ERROR.
       ELSE
         ci = "X".
     hd:BUFFER-VALUE = ci.
   END.

 END METHOD.

END CLASS.
```

c. be-PP_StkZeile



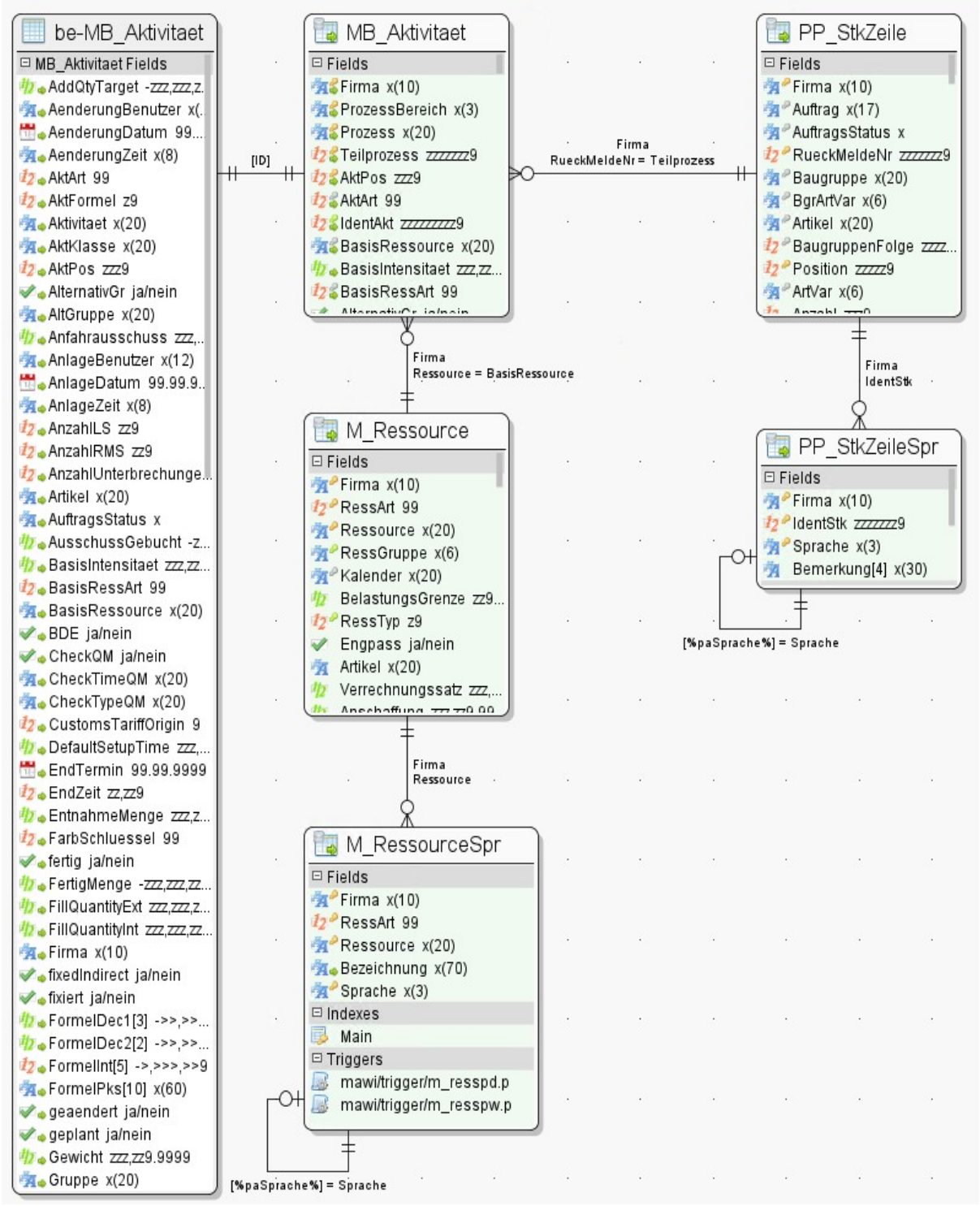
d. Backend Logik aus der be-pp_stkzeile.cls

```
// Importieren benötigter Klassen und Entitäten
USING of1.entities.be-of1.
USING proalpha.entities.be-pp_stkzeile-g.

// Definition der Klasse, die von be-pp_stkzeile-g erbt
CLASS proalpha.entities.be-pp_stkzeile INHERITS be-pp_stkzeile-g :

  // Überschreiben der Methode zum Füllen berechneter Felder
```

```
  METHOD OVERRIDE VOID mFillCalculatedFields (INPUT hbVar AS HANDLE):
    DEFINE VARIABLE hbTT      AS HANDLE NO-UNDO.
    DEFINE VARIABLE hbDB      AS HANDLE NO-UNDO.
    DEFINE VARIABLE hd        AS HANDLE NO-UNDO.
    DEFINE VARIABLE ci        AS CHARACTER NO-UNDO.
    DEFINE VARIABLE dinput    AS DECIMAL NO-UNDO.

    // Definition von Puffern für Datenbanktabellen
    DEFINE BUFFER bS_StkZeile         FOR PP_StkZeile.
    DEFINE BUFFER bS_ArtikelSpr       FOR S_ArtikelSpr.
    DEFINE BUFFER bS_MengenEinheitSpr FOR S_MengenEinheitSpr.

    // Zuweisen von Handles für temporäre und Datenbank-Tabellen
    hbTT = hbVar::hbTT.
    hbDB = hbVar::hbDB.

    // Setzen des Barcode-Codes
    hd = hbTT:BUFFER-FIELD("BarcodeCode") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "000" + hbDB::RueckMeldeNr + hbDB::StructCode.

    // Generieren und Speichern des Barcode-Bildes über einen Shell Command
    hd = hbTT:BUFFER-FIELD("BarcodeMaterialBild") NO-ERROR.
    IF VALID-HANDLE(hd) THEN
    DO:
      DEFINE VARIABLE cUri          AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cFilePath     AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cCommand      AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cbarcodeValue AS CHARACTER NO-UNDO.
      DEFINE VARIABLE lFileExists   AS LOGICAL NO-UNDO.

      // Erstellen des Barcode-Wertes und der URL für die Barcode-Generierung
      cbarcodeValue = "000" + hbDB::RueckMeldeNr + hbDB::StructCode.
      cUri          = "https://barcode.tec-it.com/barcode.ashx?data=" + cbarcodeValue +
"`&code=Code128`&translate-
esc=off`&dpi=94`&modulewidth=0.19`&height=6`&unit=Fit`&imagetype=png`&download=true`&hidehrt=t
rue".
      cFilePath     = SUBSTITUTE("e:/proalpha/pa-de-9/demo/zv/pic/" + "&1", cBarcodeValue +
"Barcode" + ".png").

      // Überprüfen, ob die Datei bereits existiert
      lFileExists   = SEARCH(cFilePath) <> ?.

      // Wenn die Datei nicht existiert, herunterladen und speichern
      IF NOT lFileExists THEN
      DO:
        cCommand = SUBSTITUTE('powershell -Command Invoke-WebRequest -Uri "&1" -OutFile "&2"',
cUri, cFilePath).
        OS-COMMAND SILENT VALUE(cCommand).
      END.

      hd:BUFFER-VALUE = cFilePath.
    END.

    // Setzen der Kurzbezeichnung für die Mengeneinheit
    hd = hbTT:BUFFER-FIELD("KurzBezeichnungMenge") NO-ERROR.
    IF VALID-HANDLE(hd) THEN DO:
      FIND FIRST bS_MengenEinheitSpr
        WHERE bS_MengenEinheitSpr.Firma           = ""
          AND bS_MengenEinheitSpr.MengenEinheit   = hbDB::MengenEinheit
          AND bS_MengenEinheitSpr.Sprache         = "D"
          NO-LOCK.

       IF AVAILABLE bS_MengenEinheitSpr THEN
           ci = bS_MengenEinheitSpr.KurzBez.
       ELSE
          ci = "X".
       hd:BUFFER-VALUE = ci.
    END.

    // Setzen der Materialbedarfsmenge
    hd = hbTT:BUFFER-FIELD("MaterialBedarfsMenge") NO-ERROR.
    IF hbDB::Mengeneinheit = 11 OR hbDB::Mengeneinheit = 10 THEN DO:
      IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = hbDB::Bedarfsmenge + ",000".
    END.
    ELSE
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = hbDB::Bedarfsmenge.

    // Setzen der Artikelbezeichnung
    hd = hbTT:BUFFER-FIELD("BezeichnungArtikelSpr") NO-ERROR.
    IF VALID-HANDLE(hd) THEN DO:
```

```
      FIND FIRST bS_ArtikelSpr
        WHERE bS_ArtikelSpr.Firma    = hbDB::Firma
          AND bS_ArtikelSpr.Artikel = hbDB::Artikel
          AND bS_ArtikelSpr.Sprache = "D"
          NO-LOCK NO-ERROR.

       IF AVAILABLE bS_ArtikelSpr THEN
         ci = bS_ArtikelSpr.Bezeichnung[1].
        ELSE
          ci = "X".
      hd:BUFFER-VALUE = ci.
    END.

    END.
END CLASS.
```

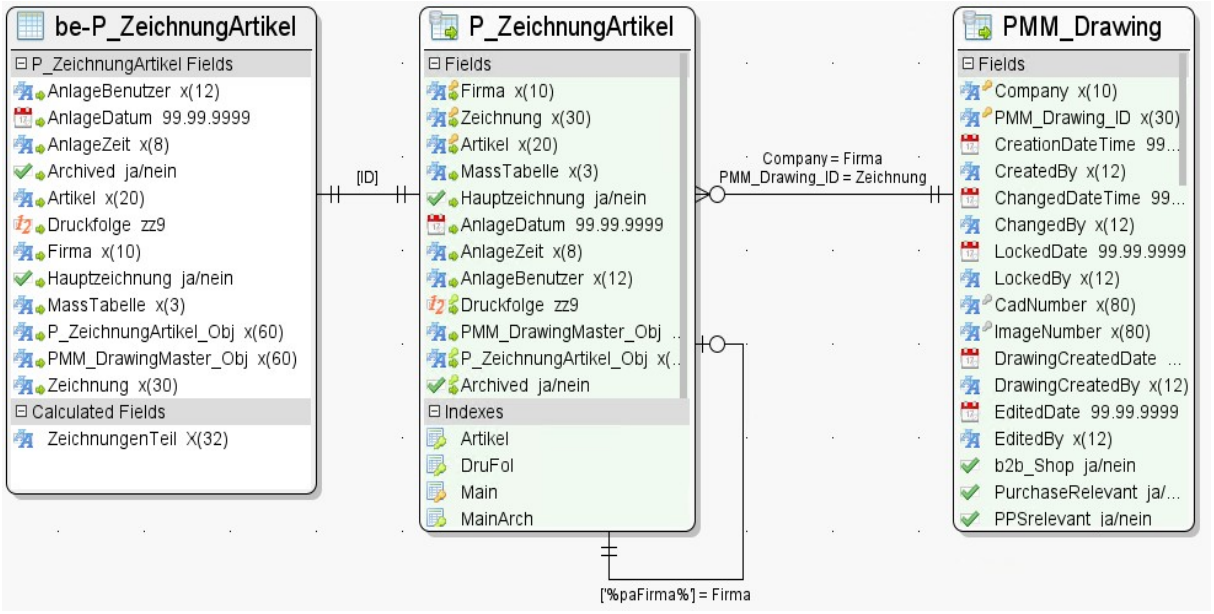
e. be-MB_Aktivitaet



f. Backend Logik aus der be-mb_aktivitaet.cls

```
// Importieren benötigter Klassen und Entitäten
USING of1.entities.be-of1.
USING proalpha.entities.be-mb_aktivitaet-g.

// Definition der Klasse, die von be-mb_aktivitaet-g erbt
CLASS proalpha.entities.be-mb_aktivitaet INHERITS be-mb_aktivitaet-g :

  // Überschreiben der Methode zum Füllen berechneter Felder
  METHOD OVERRIDE VOID mFillCalculatedFields (INPUT hbVar AS HANDLE):
    DEFINE VARIABLE hbTT AS HANDLE NO-UNDO.
    DEFINE VARIABLE hbDB AS HANDLE NO-UNDO.
    DEFINE VARIABLE hd   AS HANDLE NO-UNDO.

    // Definition eines Puffers für die Datenbanktabelle
    DEFINE BUFFER bMB_Aktivitaet FOR MB_Aktivitaet.

    // Zuweisen von Handles für temporäre und Datenbank-Tabellen
    hbTT = hbVar::hbTT.
    hbDB = hbVar::hbDB.

    // Berechnen und Setzen der Gesamtsollzeit
    hd = hbTT:BUFFER-FIELD("SollZeitGesamt") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = hbDB::soll_tr + hbDB::soll_te.

    // Setzen der Zeiteinheit auf "Min"
    hd = hbTT:BUFFER-FIELD("MinZeitEinheit") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "Min".

    // Generieren des Barcodes für Arbeitsfolgen
    hd = hbTT:BUFFER-FIELD("BarcodeArbeitsFolgen") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = "000" + hbDB::Teilprozess + hbDB::AktArt +
hbDB::AktPos.

    // Generieren und Speichern des Barcode-Bildes für Arbeitsfolgen
    hd = hbTT:BUFFER-FIELD("BarcodeFolgenBild") NO-ERROR.
    IF VALID-HANDLE(hd) THEN
    DO:
      DEFINE VARIABLE cUri          AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cFilePath     AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cCommand      AS CHARACTER NO-UNDO.
      DEFINE VARIABLE cbarcodeValue AS CHARACTER NO-UNDO.
      DEFINE VARIABLE lFileExists   AS LOGICAL NO-UNDO.

      // Erstellen des Barcode-Wertes und der URL für die Barcode-Generierung
      cbarcodeValue = "000" + hbDB::Teilprozess + hbDB::AktArt + hbDB::AktPos.
      cUri          = "https://barcode.tec-it.com/barcode.ashx?data=" + cbarcodeValue +
"`&code=Code128`&translate-
esc=off`&dpi=94`&modulewidth=0.19`&height=6`&unit=Fit`&imagetype=png`&download=true`&hidehrt=t
rue".
      cFilePath     = SUBSTITUTE("e:/proalpha/pa-de-9/demo/zv/pic/" + "&1", cBarcodeValue +
"Barcode" + ".png").

      // Überprüfen, ob die Datei bereits existiert
      lFileExists   = SEARCH(cFilePath) <> ?.

      // Wenn die Datei nicht existiert, herunterladen und speichern
      IF NOT lFileExists THEN
      DO:
        cCommand = SUBSTITUTE('powershell -Command Invoke-WebRequest -Uri "&1" -OutFile "&2"',
cUri, cFilePath).
        OS-COMMAND SILENT VALUE(cCommand).
      END.

      hd:BUFFER-VALUE = cFilePath.
    END.

    // Setzen des Rückmeldungstextes basierend auf dem Rückmeldungsstatus
    hd = hbTT:BUFFER-FIELD("RueckmeldungText") NO-ERROR.
    IF VALID-HANDLE(hd) THEN hd:BUFFER-VALUE = IF hbDB::Rueckmeldung THEN "Rückmeldung
erforderlich" ELSE "Rückmeldung nicht erforderlich".

  END METHOD.

END CLASS.
```

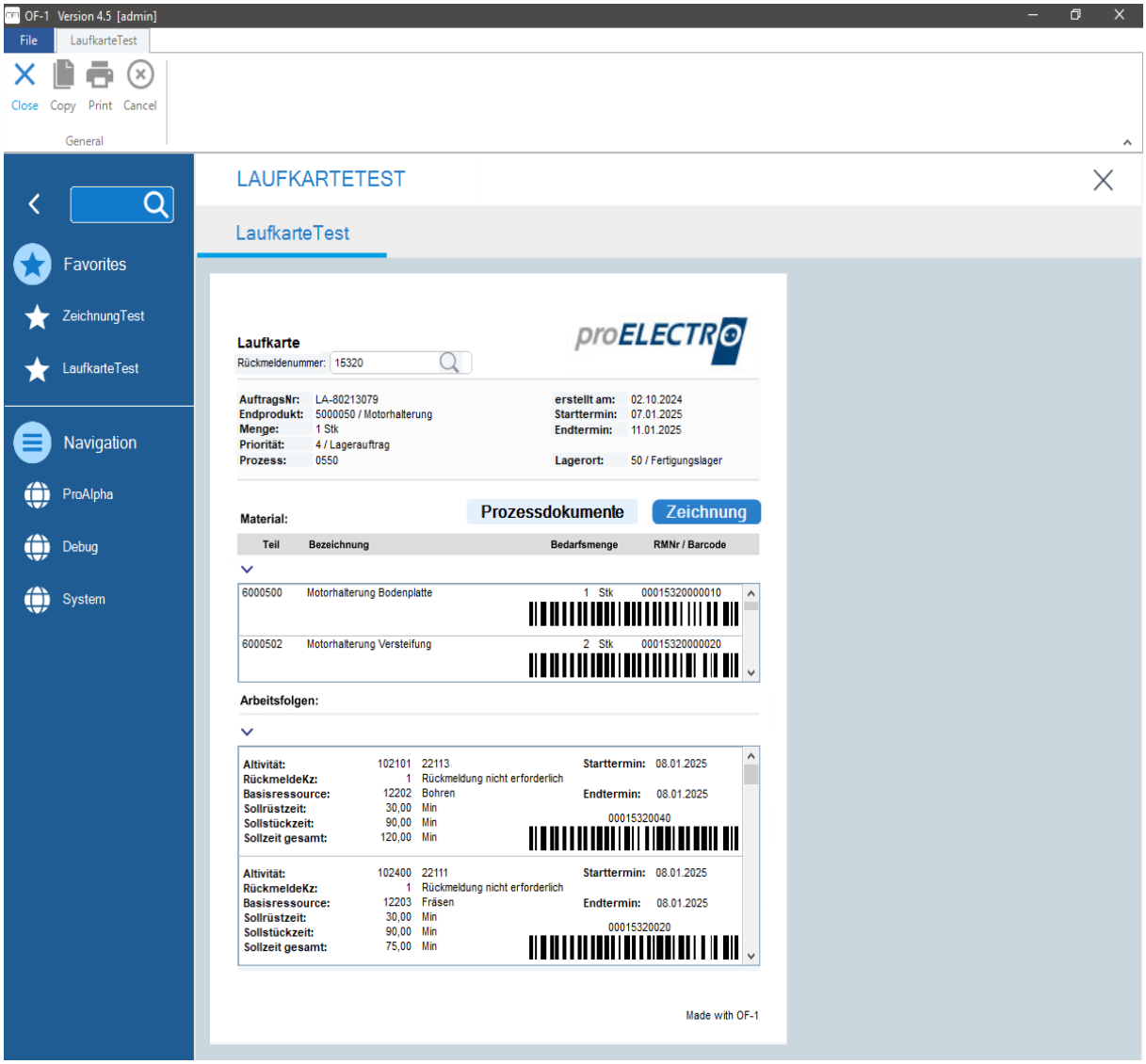
g. be-P_ZeichnungArtikel



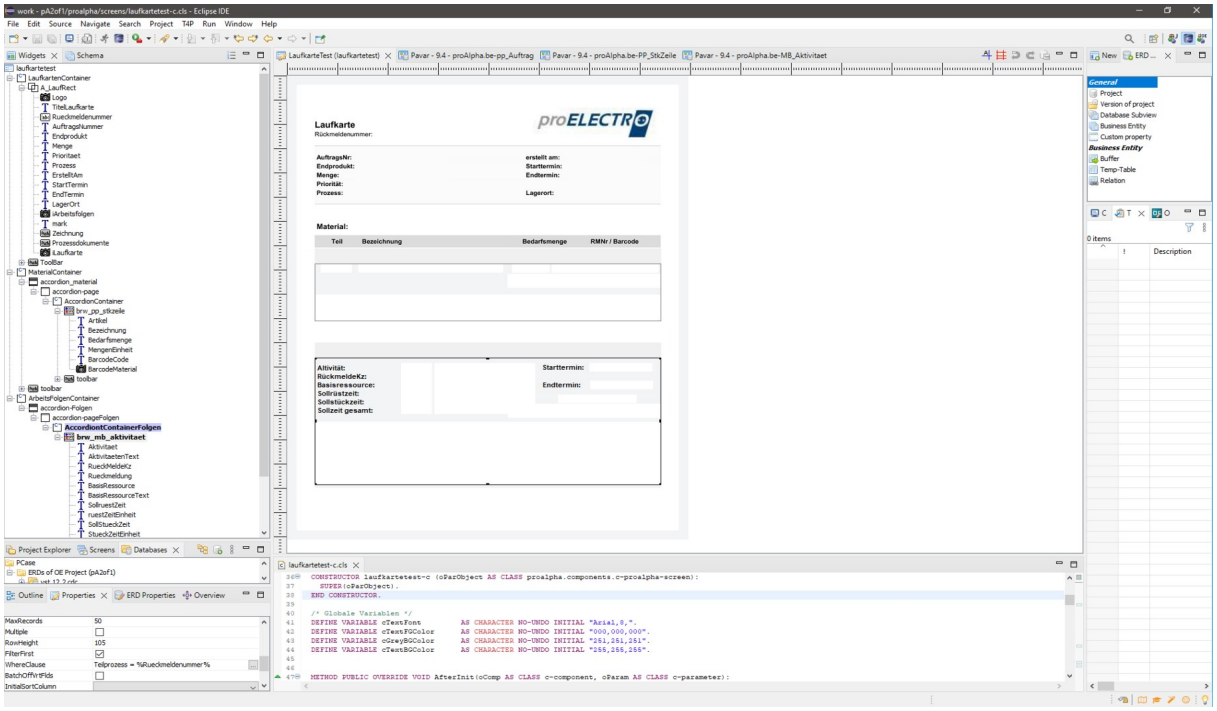
h. Backend Logik aus der be-p_zeichnungsartikel.cls

```
USING of1.entities.be-of1.
USING proalpha.entities.be-p_zeichnungartikel-g.

// Definition der Klasse, die von be-p_zeichnungartikel-g erbt
CLASS proalpha.entities.be-p_zeichnungartikel INHERITS be-p_zeichnungartikel-g :

  // Überschreiben der Methode zum Füllen berechneter Felder
  METHOD OVERRIDE VOID mFillCalculatedFields (INPUT hbVar AS HANDLE):
    DEFINE VARIABLE hbTT AS HANDLE NO-UNDO.
    DEFINE VARIABLE hbDB AS HANDLE NO-UNDO.
    DEFINE VARIABLE hd   AS HANDLE NO-UNDO.
    DEFINE VARIABLE ci   AS CHARACTER NO-UNDO.

    // Definition von Puffern für Datenbanktabellen
    DEFINE BUFFER bP_ZeichnungArtikel FOR P_ZeichnungArtikel.
    DEFINE BUFFER bPMM_Drawing        FOR PMM_Drawing.

    // Zuweisen von Handles für temporäre Datenbank-Tabellen
    hbTT = hbVar::hbTT.
    hbDB = hbVar::hbDB.

    hd = hbTT:BUFFER-FIELD("CadNumberPath") NO-ERROR.
    IF VALID-HANDLE(hd) THEN DO:

      // Suchen der zugehörigen Zeichnung
      FIND FIRST bPMM_Drawing
          WHERE bPMM_Drawing.Company        = hbDB::Firma
            AND bPMM_Drawing.PMM_Drawing_ID = hbDB::Zeichnung
          NO-LOCK.

          // Setzen der CAD-Nummer (Pfad zur Technischen Zeichnung) oder "X" wenn nicht
verfügbar
          // tauschen der .dwg Endung im String mit .png "Vorläufig"
          IF AVAILABLE bPMM_Drawing THEN
           ci = "E:/proalpha/pa-de-9/demo/zv/cad/" + REPLACE(bPMM_Drawing.CadNumber, ".dwg",
".png").
           ELSE
            ci = "X".
       hd:BUFFER-VALUE = ci.
    END.

  END METHOD.

END CLASS.
```

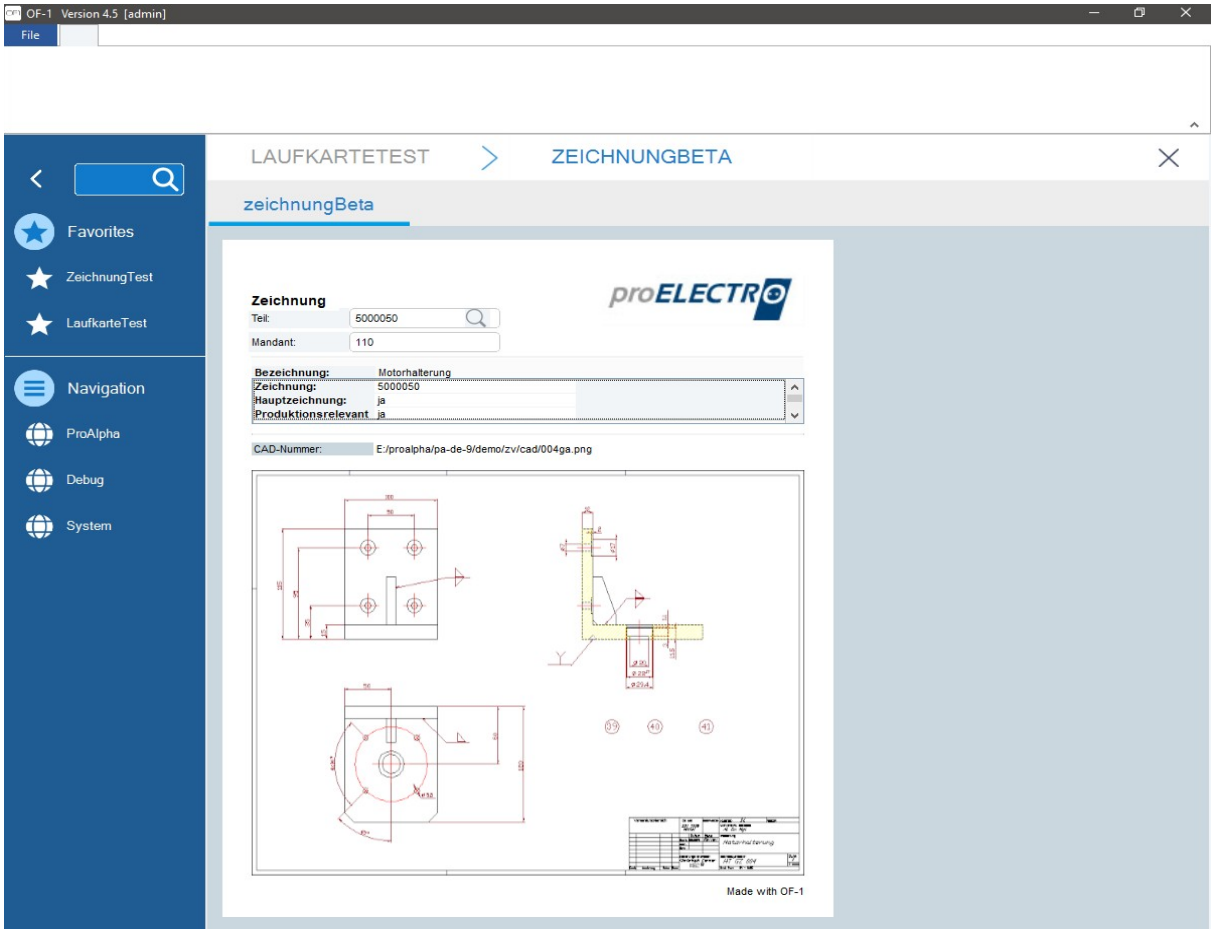
E. Benutzeroberfläche Laufkarte



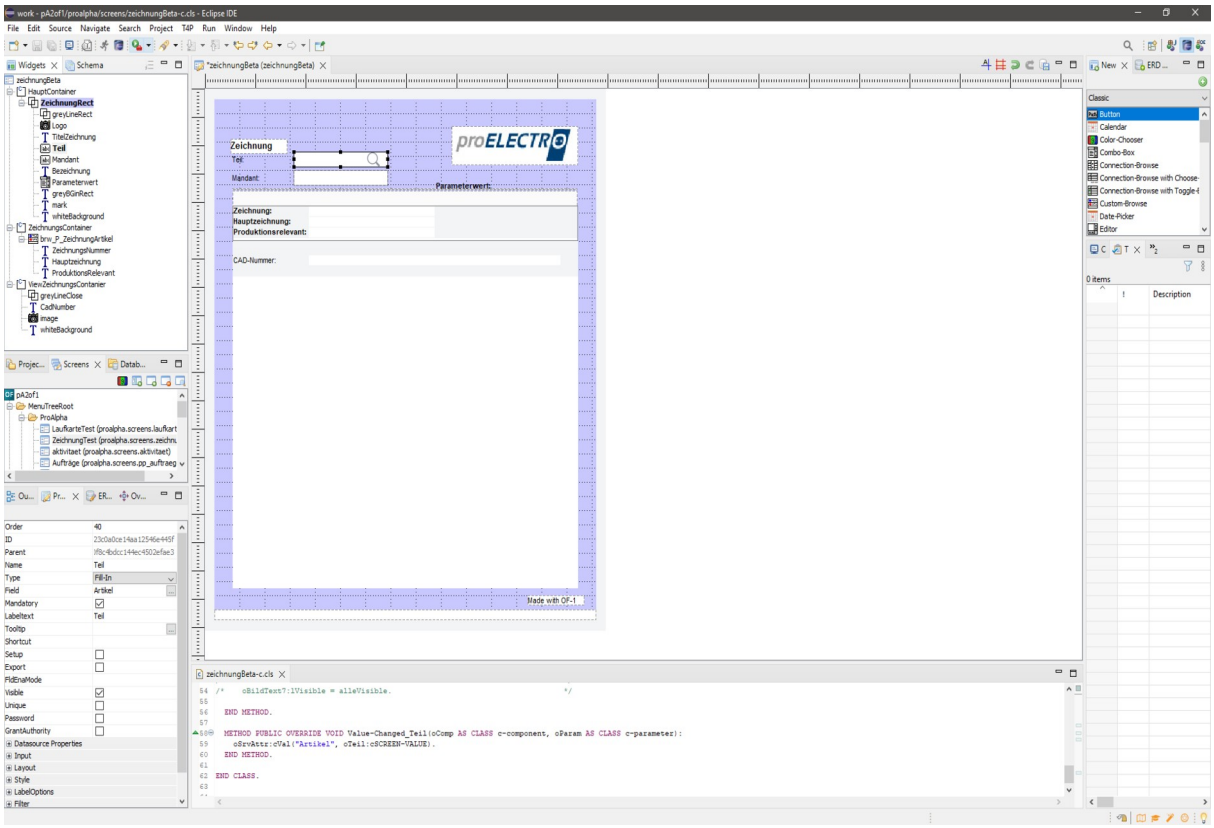
F. Designer Ansicht Laufkarte



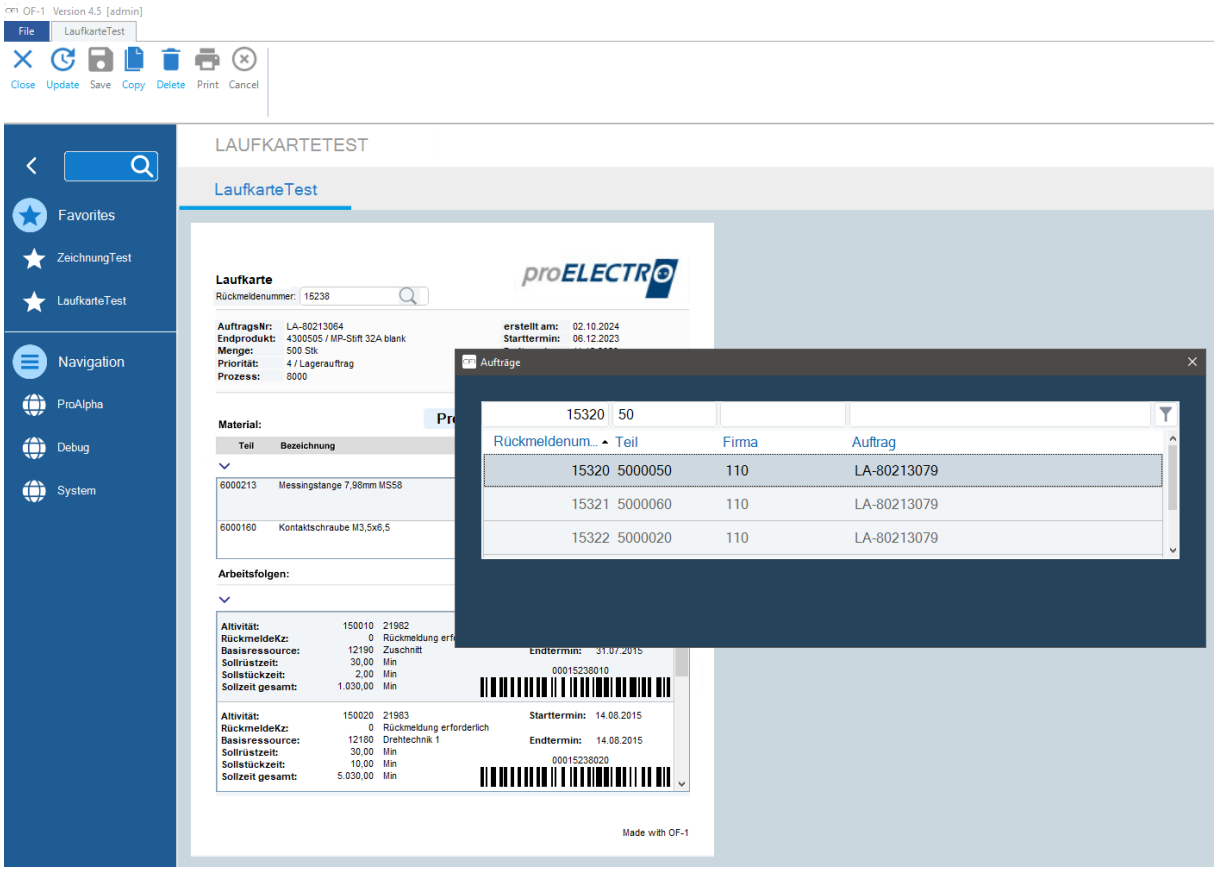
G. Benutzeroberfläche Zeichnung



H. Designer Ansicht Zeichnung

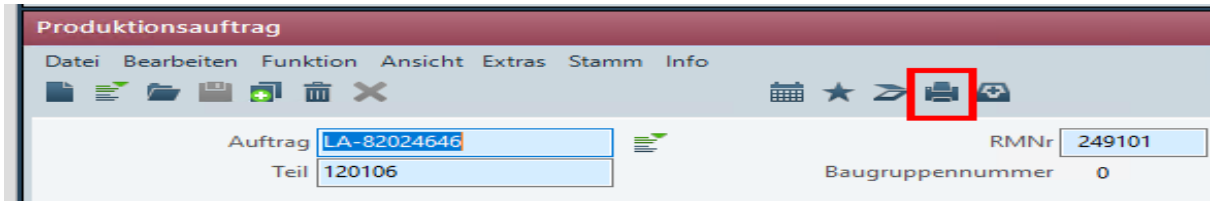


I. SubMaskeAufträge

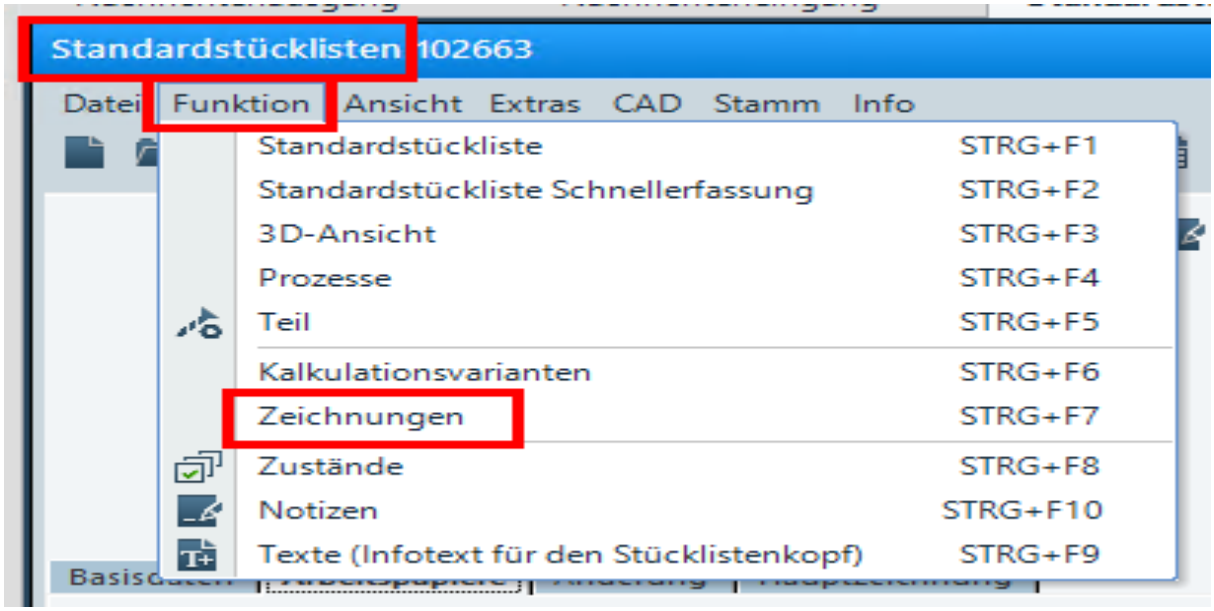


J. Screenshots

a. Laufkartenmaske



b. Zeichnungsmaske



c. Prozessdatenblattmaske