Abschlussprüfung Dezember 2024

Software Developer (IHK)
Kurs [redacted]

Dokumentation zur Projektarbeit

Data Cleaning Tool

Entwicklung einer GUI-basierten Datenbereinigungs- und Transformationspipeline in Python

Abgabedatum: [redacted] den 22.11.2024

Prüfungsbewerber:

Christopher Haase

[redacted]

Weiterbildungsplattform:

didaris

Tutor: Erwin Reichel

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CRM ..... *Customer Relation Management*
CSV ..... *Comma Separated Values*
EMEA ..... *Europe, Middle East, Africa*
EXE ..... *Executable File (ausführbare Datei)*
GUI ..... *Graphical User Interface*
IHK ..... *Industrie- und Handelskammer*
IT ..... *Informationstechnologie*
PAP ..... *Programmablaufplan*
USB ..... *Universal Serial Bus*
VID ..... *Vertriebsinnendienst*

Glossar

CustomTkinter ..... *Erweiterung von Tkinter für moderne Benutzeroberflächen*

Matplotlib .. .. *Python-Bibliothek zur Erstellung von Diagrammen*

OpenPyXL ..... *Python-Bibliothek zum Arbeiten mit Excel-Dateien*

Pandas .... *Python-Bibliothek zur Datenmanipulation und -analyse*

Tkinter .... *Standardbibliothek für grafische Benutzeroberflächen in Python*

1 Einleitung

In der heutigen Zeit der zunehmenden Digitalisierung spielt die Automatisierung von Geschäftsprozessen eine immer größere Rolle. Vor allem im Bereich der Datenaufbereitung und -analyse sind Unternehmen zunehmend auf präzise und effiziente Softwarelösungen angewiesen, um fundierte Entscheidungen treffen zu können. Im Rahmen des Zertifikatslehrgangs „Software Developer (IHK)“ entstand die Idee, eine Software zu entwickeln, die diesen Bedarf adressiert.

1.1 Projektbeschreibung

Die Anwendung „Data Cleaning Tool“ wurde entwickelt, um Rohdaten (Vertriebskennzahlen) automatisiert zu bereinigen und in ein Sales Reporting zu transformieren. Der Fokus liegt auf dem Bereich Smartphone-Vertrieb. Der Hauptnutzer des Tools ist der Vertriebsinnendienst (VID), der durch die automatisierte Bereinigung von Vertriebsdaten deutlich entlastet wird. Anstelle manueller, zeitintensiver Bereinigungen können die Mitarbeiter unbereinigte Daten in Form von CSV- oder Excel-Dateien hochladen und durch das Tool bereinigen lassen. Nach der Bereinigung wird ein finales Excel-Dokument generiert, das eine tabellarische Übersicht sowie Umsatzdiagramme enthält. Dies stellt sicher, dass die Datenqualität für Berichte und Analysen stets gewährleistet ist.

1.2 Projektumfeld

Das Projekt bezieht sich auf den Bereich der Vertriebsdatenaufbereitung, speziell im Bereich des Smartphone-Vertriebs. Der Vertriebsinnendienst spielt dabei eine zentrale Rolle, da er für die Erstellung und Bereitstellung von Reports verantwortlich ist. In der heutigen Geschäftswelt sind effiziente und zuverlässige Datenverarbeitungsprozesse entscheidend für den Erfolg eines Unternehmens. Fehlerhafte oder unvollständige Daten können zu falschen Entscheidungen führen. Das „Data Cleaning Tool“ ermöglicht dem Vertriebsinnendienst eine effiziente und automatisierte Bereinigung und Aufbereitung von Daten, was besonders in Unternehmen mit hohem Datenaufkommen relevant ist, da manuelle Prozesse häufig fehleranfällig und zeitaufwendig sind.

1.3 Ausgangssituation

Im bisherigen Prozess erfolgt die Bereinigung von Vertriebsdaten manuell, was sehr zeitaufwendig und fehleranfällig ist. Der VID-Mitarbeiter exportiert Daten aus CRM-Systemen und

bereitet diese in Excel- oder CSV-Dateien auf. Vor der Analyse müssen die Rohdaten manuell bereinigt und aufbereitet werden, da häufig Fehler wie doppelte Einträge, falsche Datumsformate oder fehlende Informationen vorhanden sind. Dieser manuelle Prozess ist nicht nur ineffizient, sondern birgt auch ein hohes Risiko für menschliche Fehler, die zu falschen Analysen und Entscheidungen führen können. Durch die Einführung des „Data Cleaning Tools“ soll dieser Prozess deutlich vereinfacht und automatisiert werden.

1.4 Projektziel

Ziel des Projekts ist es, eine benutzerfreundliche Anwendung zu entwickeln, die den Vertriebsinnendienst bei der Datenbereinigung automatisiert unterstützt. Das „Data Cleaning Tool“ soll sicherstellen, dass fehlerhafte Daten wie doppelte Einträge, fehlende Regionen oder falsche Datumsformate automatisch erkannt und korrigiert werden. Durch diese Automatisierung wird nicht nur die Qualität der Daten verbessert, sondern auch die Effizienz des Vertriebsinnendienstes gesteigert, da manuelle Arbeitsschritte entfallen. Am Ende des Bereinigungsprozesses wird ein konsistentes und korrektes Datenformat für Berichte und Analysen geliefert, das eine fundierte Entscheidungsfindung auf Basis qualitativ hochwertiger Daten ermöglicht.

2 Projektplanung

In der Projektplanung werden die einzelnen Schritte zur Umsetzung der Software definiert und die benötigten Ressourcen sowie der Entwicklungsprozess beschrieben.

2.1 Vorgehensmodell

Für die Umsetzung des Projekts wurde das Wasserfallmodell gewählt. Dieses Vorgehensmodell durchläuft die einzelnen Phasen des Projekts in einer festen Reihenfolge, was eine strukturierte und klare Umsetzung ermöglicht. Zu Beginn steht die Analyse der Anforderungen, gefolgt vom Entwurf und der Implementierung. Nach Abschluss dieser Phasen erfolgt das Testen der Anwendung, bevor sie final eingesetzt und gewartet wird.

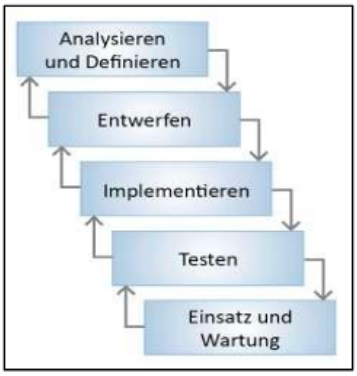

Abbildung 1: Wasserfallmodell

Das Wasserfallmodell wurde gewählt, da die strikte Reihenfolge der Phasen sicherstellt, dass die Anforderungen klar definiert und getestet werden, bevor das Projekt in den produktiven Einsatz geht.

2.2 Zeitliche Planung

Für die Bearbeitung des Projektes wurde vom Lehrgangsveranstalter ein Zeitumfang von 64 bis höchstens 80 Stunden vorgegeben.
Die einzelnen Projektphasen sind wie folgt zeitlich geplant:

Projektphase	Geplante Zeit
Analyse	2 h
Entwurf	4,5 h
Implementierung und Testen	34 h
Deployment	1 h
Dokumentation	34 h
Gesamt	**75,5 h**

Tabelle 1: Zeitplanung

Die Implementierungsphase nimmt den größten Teil der Zeit in Anspruch, da hier die tatsächliche Entwicklung des Programms stattfindet. Die Dokumentation wurde ebenfalls als wichtiger Bestandteil des Projekts mit einem großzügigen Zeitrahmen berücksichtigt.

2.3 Personal-, Sachmittel- und Kostenplanung

Für die erfolgreiche Umsetzung des Projektes ist neben der zeitlichen Planung auch eine detaillierte Planung der benötigten Sachmittel und des Personaleinsatzes erforderlich. Da es sich um ein privates Projekt handelt, werden hauptsächlich kostenlose Open-Source-Lösungen genutzt, um zusätzliche Kosten zu vermeiden. Es entstehen keine Personalkosten, da die gesamte Entwicklung eigenständig erfolgt. Bei einer realen Beauftragung müssten jedoch sowohl der Einsatz von Software- und Hardware-Ressourcen als auch die Personalkosten berücksichtigt werden. Eine hypothetische Kostenaufstellung, die solche Szenarien abdeckt, findet sich weiter unten.

Die im Projekt eingesetzten Sachmittel umfassen:

Hardware:

- Lenovo ThinkPad Intel Core i5-Prozessor, 16 GB RAM, 500 GB SSD Festplatte

- Arbeitsplatz mit zwei externen Monitoren, Tastatur und Maus, im Arbeitszimmer zuhause

Software:

- Windows 11 Pro 64-Bit Betriebssystem
- Thonny 4.1.4
- PyCharm 2024.1.1 Community Edition (IDE)
- Python 3.12 Interpreter
- Matplotlib und Pandas für die Datenvisualisierung und -verarbeitung
- Microsoft Excel für die Ausgabe der bereinigten Daten
- Microsoft Word für die Erstellung der Projektdokumentation

In einem professionellen Umfeld würde eine Entwicklung dieser Art sowohl Personalkosten als auch weitere Kosten für Lizenzen und Ressourcen nach sich ziehen. Die nachfolgende Kostenkalkulation gibt einen Einblick in die potenziellen Aufwendungen bei einer internen Beauftragung. Es wird von einem einheitlichen Bruttostundensatz von 35 Euro für alle Projektbeteiligten und einer Pauschale von 21 Euro für Personalnebenkosten ausgegangen, welche Versicherungen, Arbeitsplatzkosten und ähnliche Faktoren abdecken.

Anwendung	Anzahl Mitarbeitende	Zeit	Bruttoarbeitsentgelt	Personalnebenkosten	Gesamt
IT (Softwareentwicklung)	1	70 h	2.450 €	1.470 €	3.920 €
Vertriebsinnendienstabteilung (Fachgespräch, Abnahme, etc.)	2	4 h	280 €	168 €	448 €
Kosten gesamt					**4.368 €**

Tabelle 2: Kostenkalkulation (hypothetisch)

Im Rahmen einer tatsächlichen Projektbeauftragung innerhalb eines Unternehmens würden voraussichtlich Kosten von etwa 4.368 Euro anfallen. Auch wenn eine Berechnung der Amortisation unter klassischen Gesichtspunkten möglich ist, und diese theoretisch nach etwa 13,78 Jahren erreicht würde, steht die Amortisationsrechnung hier nicht im Vordergrund.

Ein wesentlicher Grund für die lange Amortisationsdauer liegt darin, dass das Reporting auf monatlicher Basis gesichtet und bewertet wird. Würde das Tool häufiger genutzt werden, zum Beispiel für wöchentliche oder sogar tägliche Auswertungen, so würde sich die Amortisationsdauer deutlich verkürzen.
Vielmehr liegt der wesentliche Vorteil der Einführung der Software nicht in einer direkten, schnellen Rückführung der Kosten durch monetäre Einsparungen, sondern in der erheblichen Effizienzsteigerung, Fehlervermeidung und Verbesserung der Datenqualität. Die Vermeidung strategischer Fehlentscheidungen, die auf falschen Daten basieren könnten, und die damit verbundenen potenziell weitaus höheren Kosten, machen die Nutzung der Software deutlich rentabler. Das primäre Ziel ist es, den manuellen Aufwand zu reduzieren und dadurch die Qualität und Geschwindigkeit der Prozesse nachhaltig zu verbessern. Insofern übersteigt der Nutzen des Projekts die reine Amortisationsberechnung. Siehe dazu A1 Amortisationsberechnung.

3 Analysephase

Die Analyse bildet die Grundlage für die nachfolgende Entwurfs- und Implementierungsphase. Nach der Erhebung des Ist-Zustands und der Analyse der derzeitigen Herausforderungen, die durch die entwickelte Anwendung behoben werden sollen, wird der Soll-Prozess definiert.

3.1 Ist-Analyse

Derzeit erfolgt die Bereinigung von Vertriebsdaten manuell durch den Vertriebsinnendienst, was fehleranfällig und zeitaufwendig ist. Die Daten werden oft aus CRM-Systemen exportiert und liegen in CSV- oder Excel-Formaten vor. Diese müssen vor der Analyse manuell aufbereitet werden, da häufige Probleme wie doppelte Einträge, inkorrekte Datumsformate und fehlende Angaben vorhanden sind. Dieser manuelle Prozess führt zu Verzögerungen und birgt das Risiko, dass menschliche Fehler übersehen werden. Dies kann die Genauigkeit der Analysen beeinträchtigen und zu fehlerhaften geschäftlichen Entscheidungen führen.
Durch das Data Cleaning Tool wird dieser manuelle Aufwand erheblich reduziert, indem Fehler automatisiert erkannt und behoben werden. Dadurch wird die Datenqualität deutlich verbessert, der Bereinigungsprozess beschleunigt und die Effizienz gesteigert. Das Tool ermöglicht es, konsistente und fehlerfreie Daten für wichtige unternehmerische Entscheidungen zu nutzen, ohne dass langwierige manuelle Korrekturen erforderlich sind. Eine grafische Abbildung des Ist-Prozesses ist im Anhang unter A2 Ist-Prozess zu finden.

3.2 Soll-Prozess

Das Ziel des Data Cleaning Tools ist es, den Prozess der Datenbereinigung zu automatisieren und Fehler, die in den exportierten Rohdaten aus CRM-Systemen auftreten, systematisch zu korrigieren. Das Tool stellt sicher, dass fehlerhafte Daten wie doppelte Einträge, falsche Datumsformate und fehlende Regionen automatisch bereinigt werden. Nach der Bereinigung werden die Daten in einem standardisierten Format ausgegeben und stehen für weitere Analysen zur Verfügung. Dies spart Zeit und erhöht die Effizienz, da die manuelle Bearbeitung entfällt und wiederholbare Fehler zuverlässig erkannt und behoben werden.
Ein weiteres Ziel des Tools ist es, die Daten direkt in Berichtsform zu visualisieren, z.B. als monatliche Umsatzübersicht oder detaillierte Verkaufsberichte pro Verkäufer. Der Prozess wird vollständig automatisiert, sodass der Benutzer die Daten nur einmal hochladen muss, um am Ende des Prozesses eine bereinigte und visuell aufbereitete Datei zu erhalten. Die grafische Abbildung des Soll-Prozesses ist im Anhang unter A3 Soll-Prozess zu finden.

3.3 Anwendungsfälle

Die Anwendung unterstützt verschiedene Anwendungsfälle, die den vollständigen Bereinigungs- und Analyseprozess abbilden. Dazu gehören:

- **Laden von CSV/Excel-Rohdaten**: Der Benutzer lädt die exportierten, unbereinigten Rohdaten in die Anwendung.
- **Automatische Datenbereinigung**: Das Tool erkennt Fehler wie doppelte Einträge, falsche Datumsformate und fehlende Angaben und korrigiert diese automatisch.
- **Speichern der bereinigten Daten**: Die fehlerbereinigten Daten werden in einer neuen Excel-Datei gespeichert.
- **Erstellung von Berichten**: Nach der Bereinigung generiert das Tool automatisch Berichte und visualisiert monatliche Umsätze und Verkäuferdaten in Diagrammen.

Das zugehörige Use-Case-Diagramm ist im Anhang unter A4 Use-Case Diagramm zu finden.

4 Entwurfsphase

Um den Prozess zur Datenbereinigung effizient zu gestalten, wird eine Anwendung entwickelt, die Nutzer durch den gesamten Prozess der Datenaufbereitung führt. Sowohl die Benutzeroberfläche (GUI) als auch die Anwendungslogik werden in diesem Kapitel beschrieben.

4.1 Zielplattform

Die Anwendung wird als ausführbare .exe-Datei für Windows 11 bereitgestellt, da dieses Betriebssystem während der Entwicklung verwendet wurde. Python wurde als Programmiersprache gewählt, da es sich besonders gut für datenbezogene Aufgaben und Automatisierungen eignet. Dank der umfangreichen Bibliotheken, wie „pandas“ für die Datenverarbeitung und „openpyxl“ für den Excel-Export, bietet Python die nötige Flexibilität und Effizienz. Prinzipiell wäre auch eine Implementierung in einer anderen Programmiersprache wie Java möglich gewesen.

4.2 Entwurf der Anwendungslogik und Benutzeroberfläche

Die Kernlogik der Anwendung umfasst:

- **Datenbereinigung**: Automatische Erkennung und Korrektur fehlerhafter Datensätze (z.B. fehlende Regionen, doppelte Einträge).
- **Datenanreicherung**: Berechnung von Feldern wie Gesamtumsatz und Monatsaggregationen.
- **Berichterstellung**: Generierung von Berichten in Form von Diagrammen und Tabellen, die mit **Matplotlib** visualisiert und in eine Excel-Datei integriert werden.

Fehler werden durch Pop-up-Meldungen angezeigt, und eine schrittweise Menüführung verhindert, dass Nutzer zu schnell durch den Prozess gehen, bevor die vorherigen Schritte abgeschlossen sind. Die Hauptfunktion der Anwendung ist es, die Rohdaten zu bereinigen und anschließend in einem leicht verständlichen Bericht darzustellen.
Die Anwendung wird mit einer grafischen Benutzeroberfläche (GUI) entwickelt, um eine benutzerfreundliche und intuitive Handhabung zu gewährleisten. Die GUI wurde mit der Bibliothek **CustomTkinter** realisiert, die auf dem Tkinter-Framework aufbaut und erweiterte Design- und Layout-Optionen bietet. Das Hauptfenster der Anwendung bietet eine einfache Menüführung, bei der die Nutzer die Rohdaten im CSV- oder Excel-Format laden und bereinigen können. Eine Fortschrittsanzeige gibt Aufschluss über den Status des Bereinigungsprozesses

4.3 Programmablauf

Der Programmablaufplan (PAP) beschreibt den strukturierten Ablauf der Anwendung von der Eingabe der Rohdaten bis zur finalen Datenverarbeitung und Berichterstellung. Der Prozess beginnt mit dem Hochladen der Rohdaten (CSV oder Excel) durch den Benutzer. Im Anschluss erfolgt die automatische Verarbeitung, in der Fehler wie doppelte Einträge, falsche Datums-

formate oder fehlende Regionen identifiziert und korrigiert werden. Nach Abschluss der Bereinigung werden die bereinigten Daten in einer Excel-Datei gespeichert, die zusätzlich grafische Auswertungen wie Umsatzdiagramme enthält.
Zur Verdeutlichung der Abläufe wurde ein detaillierter **Programmablaufplan** erstellt, der im Anhang unter A5 Programmablaufplan zu finden ist. Die einzelnen Funktionen der Anwendung, wie die Datenprüfung, Bereinigung und Berichterstellung, sind einem **Struktogramm** abgebildet, die im Anhang A6 Struktogramm dargestellt werden.

4.4 Datenmodell

Das Datenmodell der Anwendung basiert auf der Verarbeitung von CSV- oder Excel-Rohdaten, die Vertriebsdaten enthalten. Die wesentlichen Datenfelder umfassen:

- **Datum**: Verkaufsdatum
- **Verkäufer**: Name des Verkäufers
- **Region**: Verkaufsregion (z.B. EMEA, AMERICAS, ASIA)
- **Produkt**: Name des Produkts
- **Verkaufte Menge**: Anzahl der verkauften Einheiten
- **Umsatz pro Einheit**: Einzelpreis des Produkts
- **Gesamtumsatz**: Verkaufte Menge * Umsatz pro Einheit
- **Kommentar**: Hinweise auf fehlerhafte Daten (z.B. falsches Datumsformat, fehlende Region, doppelte Einträge)

Die Bereinigung der Daten erfolgt durch das Tool, das doppelte Einträge, fehlende Regionen oder falsche Datumsformate automatisch erkennt und korrigiert. Nach der Bereinigung werden die bereinigten Daten in einer neuen Excel-Datei gespeichert und mit weiteren berechneten Feldern wie Gesamtumsatz und Monat angereichert.

Wichtiger Hinweis:
Das Data Cleaning Tool setzt voraus, dass die Rohdaten in der Datei „Rohdaten.csv“ unverändert bleiben. Die Struktur der Eingabedatei, einschließlich der Spaltennamen, der Reihenfolge der Spalten und der Datentypen, darf nicht verändert werden. Änderungen an der CSV-Datei außerhalb der Anwendung können die automatische Bereinigung und Transformation beeinträchtigen und zu Fehlfunktionen führen.

4.5 Geschäftslogik

Die Geschäftslogik der erstellten Anwendung beschreibt die wesentlichen Schritte zur Bereinigung und Analyse der Verkaufsdaten. Ein zentraler Bestandteil der Logik ist die Automatisierung der Datenbereinigung, um fehlerhafte Datensätze zu identifizieren und zu korrigieren. Diese Bereinigungslogik umfasst:

- **Datenbereinigung**: Automatische Erkennung und Korrektur von fehlerhaften oder fehlenden Daten (z.B. fehlende Regionen, falsches Datumsformat, doppelte Einträge).
- **Datenanreicherung**: Berechnung von Feldern wie „Gesamtumsatz“ und Aggregation von Verkäufen auf monatlicher Basis.
- **Validierung**: Überprüfung der Datenintegrität nach der Bereinigung.

Die Anwendung verarbeitet die Rohdaten und erzeugt eine bereinigte Excel-Datei, die als Grundlage für fundierte Entscheidungen in Vertrieb und Management dient.

5 Implementierungsphase

Die Implementierung beschreibt die technische Umsetzung der Anwendung. Im Folgenden wird beschrieben, wie die Anforderungen programmatisch umgesetzt wurden.

5.1 Implementierung der Anwendungslogik und Benutzeroberfläche

Anwendungslogik:

Die gesamte Anwendungslogik ist in der Klasse **DataCleaningApp** zentralisiert. Diese Klasse bündelt alle Funktionen, die für die Datenbereinigung, -analyse und das Erstellen von Berichten notwendig sind. Durch diese Struktur ist es möglich, die Logik und den Benutzerfluss effizient zu verwalten.

Ein zentraler Bestandteil der Anwendungslogik ist die Funktion **clean_data()**, die alle Schritte zur Bereinigung der Daten in einem klaren, strukturierten Ablauf durchführt:

- **Erkennung von doppelten Einträgen:** Die Funktion verwendet Pandas' **.drop_duplicates()-Methode**, um sicherzustellen, dass keine doppelten Datensätze vorhanden sind.
- **Korrektur von Datumsformaten:** Die Funktion **parse_date()** wird auf das Datum angewendet, um sicherzustellen, dass alle Datumsangaben in einem konsistenten Format vorliegen.

- **Füllen fehlender Werte:** Fehlende Daten, wie z.B. Regionen, werden durch die Funktion **fill_region()** automatisch ergänzt, basierend auf vordefinierten Regeln und historischen Daten.

```python
def clean_data(df):
    # Bereinigung Datensatz und Korrektur Datumsformate

    # Entfernen von doppelten Einträgen, außer dem Kommentar
    df = df.drop_duplicates(
        subset=['Datum', 'Verkäufer', 'Region', 'Produkt', 'Verkaufte Menge', 'Umsatz pro Einheit', 'Gesamtumsatz'],
        keep='first'
    ).copy()
```

Abbildung 2: Entfernen doppelter Einträge

Ein besonderes Merkmal der Anwendung ist die Funktion **save_charts(),** die eine anspruchsvolle Aufgabe übernimmt:

- **Erstellung von Vergleichsbalken-Diagrammen:** Diese Funktion nutzt „**Matplotlib**“, um detaillierte Vergleichsbalkendiagramme zu erstellen, die den Umsatz pro Monat sowie pro Verkäufer darstellen. Diese Diagramme bieten eine visuelle Übersicht über die Verkaufsperformance und können für detaillierte Analysen verwendet werden.
- **Integration der Diagramme in die Excel-Datei:** Nach der Erstellung werden die Diagramme automatisch in die bereinigte Excel-Datei eingebettet. Dies geschieht mithilfe von „**OpenPyXL**“, das die Diagramme als Bilddateien speichert und an der richtigen Position in der Excel-Datei platziert.

```python
def save_charts(self, df):
    # Erstellung Balkendiagramme und Speicherung in Excel-Datei
    # Darstellung Monate auf Deutsch
    locale.setlocale(locale.LC_TIME, 'de_DE.UTF-8')

    df['Monat'] = pd.to_datetime(df['Datum']).dt.to_period('M')
    df['Gesamtumsatz'] = df['Verkaufte Menge'] * df['Umsatz pro Einheit']

    monthly_sales = df.groupby('Monat')['Gesamtumsatz'].sum()

    # Erstellung erstes Diagramm: Gesamtumsatz pro Monat
    fig, ax = plt.subplots(figsize=(10, 6))
    monthly_sales.plot(kind='bar', ax=ax)
    ax.set_xticklabels([period.strftime('%B %Y') for period in monthly_sales.index], rotation=45, ha="right")
    ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'{x:,.0f} €'))
    ax.set_title("Gesamtumsatz pro Monat")
    ax.set_xlabel("Monat")
    ax.set_ylabel("Gesamtumsatz in €")
    ax.grid(True)
```

Abbildung 3: Erstellung Balkendiagramme

Fehlerbehandlung und Stabilität durch Try, Except:

Ein entscheidender Aspekt der Anwendungslogik ist die Fehlerbehandlung mittels **Try, Except**. Dies sorgt dafür, dass kritische Abschnitte wie das Laden der Dateien, die Datenbereinigung und die Diagrammerstellung robust bleiben. Sollte ein unerwarteter Fehler auftreten, wird dieser abgefangen, und dem Benutzer wird eine verständliche Fehlermeldung präsentiert, anstatt dass die Anwendung abstürzt. Dadurch wird die Stabilität und Benutzerfreundlichkeit des Tools erheblich gesteigert.

Diese modulare Struktur innerhalb der DataCleaningApp-Klasse sorgt für eine hohe Flexibilität und Wiederverwendbarkeit der Code-Module, was zukünftige Erweiterungen oder Anpassungen an neue Datentypen erleichtert.

Grafische Benutzeroberfläche:

Die grafische Benutzeroberfläche (GUI) wurde unter Verwendung der Bibliothek **CustomTkinter** entwickelt, die auf Tkinter aufbaut, jedoch erweiterte Designoptionen bietet. Die Wahl fiel auf CustomTkinter, um dem Tool ein modernes Aussehen und eine intuitive Benutzerführung zu geben, während gleichzeitig die Funktionalität gewährleistet bleibt.

Hauptkomponenten der GUI:

- **Hauptfenster und Menüführung:** Das Hauptfenster der Anwendung bietet dem Benutzer eine einfache Navigation durch die wichtigsten Funktionen: Laden der Rohdaten, Bereinigung der Daten und Export der bereinigten Daten. Die Menüführung ist schrittweise aufgebaut, was bedeutet, dass der Benutzer den nächsten Schritt erst ausführen kann, wenn der vorherige erfolgreich abgeschlossen wurde. Dadurch wird die Fehleranfälligkeit reduziert und eine klare Struktur im Arbeitsablauf gewährleistet.
- **Buttons und Fortschrittsanzeige:** Verschiedene Buttons steuern den gesamten Prozess. Der Button „Daten laden“ ermöglicht das Einlesen der CSV- oder Excel-Datei, und der Button „Bereinigung starten“ führt den Bereinigungsprozess durch. Ein Fortschrittsbalken zeigt den Fortschritt jeder Operation an und gibt dem Benutzer visuelle Rückmeldungen über den Status der Datenverarbeitung.

```python
def _create_progress_bar(self):
    # Erstellung Fortschrittsanzeige und zugehörige Label
    progress_frame = ctk.CTkFrame(self.root, fg_color="#F2F2F7")
    progress_frame.pack(pady=10, padx=20, fill="x")

    self.progress_bar = ctk.CTkProgressBar(
        progress_frame, width=PROGRESS_BAR_WIDTH, height=20, progress_color=BUTTON_COLOR
    )
    self.progress_bar.set(0)
    self.progress_bar.pack(side="left", pady=10, padx=(20, 2))

    self.progress_label = self._create_label( text: "0%", parent=progress_frame, font_size=10, bold=True)
    self.progress_label.pack(side="right", padx=(0, 10))
```

Abbildung 4: Fortschrittsanzeige

- **Prozessführung:** Die Benutzeroberfläche leitet den Nutzer durch alle Phasen des Data Cleaning. Nach dem Laden der Rohdaten wird der Fortschritt jeder Verarbeitungsschritt durch den Fortschrittsbalken angezeigt. Dies gibt dem Benutzer die Möglichkeit, die Automatisierung zu überwachen und sicherzustellen, dass jeder Schritt erfolgreich abgeschlossen wurde.
- **Design und Usability:** Die GUI wurde nach Prinzipien der Usability gestaltet, um die Bedienung für den Endbenutzer so einfach wie möglich zu gestalten. Beispielsweise werden inaktive Buttons automatisch deaktiviert, wenn ein Schritt noch nicht abgeschlossen ist. Dies verhindert, dass der Benutzer in einen fehlerhaften Zustand gerät, und sorgt für eine benutzerfreundliche Bedienung. Farbkontraste und klare Beschriftungen der Buttons tragen dazu bei, dass auch weniger erfahrene Benutzer die Anwendung leicht bedienen können.

5.2 Testen der Anwendung

Um die Qualität der Anwendung zu gewährleisten, wurden während der gesamten Implementierungsphase kontinuierlich Tests durchgeführt. Dies beinhaltete sowohl manuelle Tests als auch automatisierte Unittests.

Schreibtischtests:

- Zur Überprüfung der Logik und des Programmablaufs wurden Schreibtischtests durchgeführt. Diese Methode diente dazu, jeden Schritt der Datenbereinigung gedanklich durchzugehen, bevor die tatsächliche Implementierung stattfand. Auf diese Weise konnten potenzielle Fehler bereits im Vorfeld erkannt und vermieden werden.
- Diese Tests erwiesen sich besonders bei der Entwicklung der komplexen Funktionen, wie der Datenvalidierung und -bereinigung, als wertvoll, da sie es ermöglichten, Schwachstellen in der Logik zu identifizieren, ohne dass der Code bereits ausgeführt werden musste.

Manuelle Tests und Debugging:

- Schon während der Implementierung wurden regelmäßig Kontrollausgaben **(print())** verwendet, um die Zwischenergebnisse der Datenbereinigung zu überprüfen. Dies half dabei, frühzeitig Fehler im Code zu identifizieren und zu beheben.
- Besonders Augenmerk lag auf der Plausibilitätsprüfung der Datenformate und der Integrität der Excel-Exporte. Hier wurde getestet, ob alle bereinigten Daten korrekt in die Excel-Dateien exportiert werden und ob die Diagramme ordnungsgemäß erstellt und eingebettet werden.

- Weitere manuelle Tests wurden durchgeführt, um die Benutzerführung und die GUI zu überprüfen. Hierbei wurde geprüft, ob die Benutzeroberfläche intuitiv bedienbar ist und ob alle Buttons und Menüelemente wie vorgesehen funktionieren.

Automatisierte Unittests:

- Für die Funktionalität der Datenbereinigung wurden Unittests implementiert. Hierbei wurden typische Fehlerfälle wie fehlende Werte, doppelte Einträge und falsche Datumsformate getestet, um sicherzustellen, dass die Bereinigungslogik korrekt funktioniert.
- Für jede wichtige Funktion der Anwendung wurde mindestens ein Testfall definiert, um sicherzustellen, dass die Datenintegrität nach der Bereinigung gewährleistet bleibt.
- Die Unittest-Datei wird zusammen mit der Programmdatei bereitgestellt.

Durch diese umfassenden Teststrategien konnte sichergestellt werden, dass die Anwendung fehlerfrei funktioniert und alle gestellten Anforderungen erfüllt.

6 Deployment**phase**

In der Deployment**phase** wird die Bereitstellung des Data Cleaning Tools beschrieben. Nach Abschluss der Entwicklung wird das Programm als ausführbare .exe-Datei für Windows bereitgestellt. Das Tool benötigt keine aufwändige Installation – die ausführbare Datei kann direkt genutzt werden.

6.1 Deployment-Schritte

Der Deployment-Prozess war einfach und erforderte keine automatisierten Tools. Die Schritte zum Deployment beinhalteten:

- **Erstellung der ausführbaren .exe-Datei**: Das Tool wurde mittels PyInstaller in eine .exe-Datei kompiliert, die auf jedem Windows-System ohne Python-Installation lauffähig ist.
- **Dateiverteilung**: Das Programm wurde so vorbereitet, dass es problemlos über Netzlaufwerke oder USB-Sticks verteilt werden kann.

Es war keine Datenmigration notwendig, da das Tool direkt auf CSV- oder Excel-Dateien angewendet wird, die im Arbeitsablauf bereitgestellt werden.

6.2 Benutzerschulung

Da das Tool eine benutzerfreundliche Oberfläche bietet, waren keine umfangreichen Schulungen erforderlich. Eine kurze Einweisung in die grundlegende Funktionsweise reichte aus, um die Nutzer mit dem Ablauf vertraut zu machen. Für detaillierte Informationen und Nachfragen steht den Nutzern eine ausführliche Benutzerdokumentation zur Verfügung. Diese enthält eine Schritt-für-Schritt-Anleitung zur Bedienung des Tools, einschließlich des Ladens der Rohdaten, der Bereinigung und des Exports der finalen Excel-Datei.

7 Dokumentation

Zusätzlich zur Projektdokumentation, die die Umsetzung des Data Cleaning Tools beschreibt, wurde eine ausführliche Benutzerdokumentation erstellt. Diese erklärt den Anwendern Schritt für Schritt die Bedienung des Tools, einschließlich des Ladens der Rohdaten, der Bereinigung sowie des Exports der bereinigten Daten und Diagramme in Excel. Diese ist im Anhang unter A7 Benutzerdokumentation zu finden.

8 Projektergebnis

Zum Abschluss des Projekts wird im Folgenden das Projektergebnis beschrieben. Dabei wird ein Soll-Ist-Abgleich vorgenommen, um die Implementierung und den tatsächlich aufgewendeten Zeitrahmen zu reflektieren. Zudem werden die während der Projektumsetzung gewonnenen Lernerfahrungen zusammengefasst. Dies ermöglicht eine abschließende Bewertung des Projekts sowie eine Betrachtung der wichtigsten Erkenntnisse und Herausforderungen, die während der Entwicklung aufgetreten sind.

8.1 Anwendung

Das Ergebnis des Projekts ist das „Data Cleaning Tool“, eine Anwendung, die es ermöglicht, unbereinigte Rohdaten (Vertriebskennzahlen) zu laden, automatisch zu bereinigen und in einem finalen Excel-Report zu exportieren. Diese Anwendung erkennt und korrigiert Fehler wie doppelte Einträge, fehlende Regionen und falsche Datumsformate. Zudem generiert die Anwendung auf Basis der bereinigten Daten Umsatzdiagramme und Berichte, die zur strategischen Entscheidungsfindung genutzt werden können.
Die grafische Benutzeroberfläche des Tools führt den Benutzer intuitiv durch den gesamten Prozess und stellt sicher, dass die wichtigsten Schritte durch Fortschrittsanzeigen und klare

Menüs unterstützt werden. Die Benutzeroberfläche ist so gestaltet, dass sie leicht bedienbar und flexibel anpassbar ist.

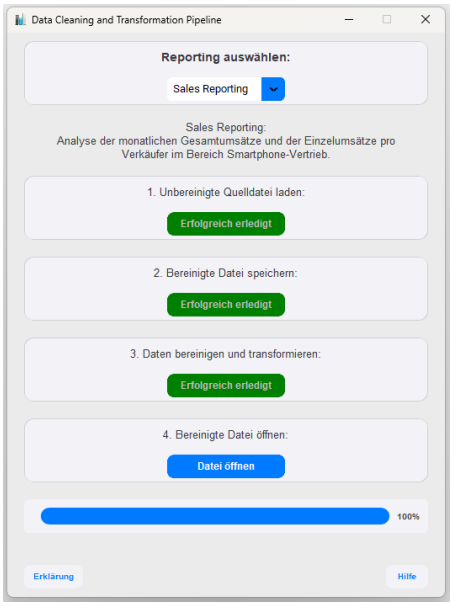

Abbildung 5: Grafische Benutzeroberfläche

Die Erzeugung der Excel-Datei erfolgt im Anschluss an die Datenbereinigung und Transformation. In dieser Excel-Datei werden die bereinigten Daten übersichtlich dargestellt, ergänzt durch detaillierte Balkendiagramme, die sowohl den Gesamtumsatz pro Monat als auch den Umsatz pro Verkäufer visualisieren. Diese Diagramme unterstützen die Entscheidungsfindung und bieten einen klaren Überblick über die Verkaufsleistung im Bereich Smartphone-Vertrieb.

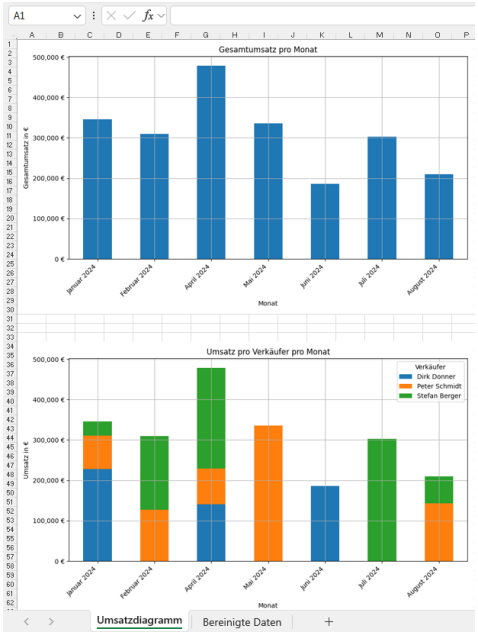

Abbildung 6: Erzeugung Excel-Datei mit Balkendiagrammen

Der gesamte Quellcode für die entwickelte Anwendung ist im Anhang unter A8 Quellcode beigefügt.

8.2 Soll-Ist-Abgleich

Die Anwendung wurde entsprechend den Projektzielen vollständig realisiert. Alle geplanten Funktionen, wie die automatisierte Datenbereinigung und die Erstellung von Reports, wurden erfolgreich implementiert und getestet. Bei der Entwicklung der Anwendung wurde sehr auf die Erfüllung der relevanten und erforderlichen Qualitätskriterien (Korrektheit, Robustheit, Zuverlässigkeit, Benutzerfreundlichkeit, Effizienz und Wartbarkeit) geachtet.
Während der Implementierung traten unerwartete Herausforderungen bei der Funktion **save_charts()** zur Erstellung des Excel-Berichts und der Visualisierung auf, da die Integration der Diagramme mehr Anpassungen erforderte, als ursprünglich geplant. Zudem führte die automatisierte Datenbereinigung, insbesondere der Umgang mit fehlenden Daten in der Funktion **clean_data()** zur Standardisierung von Datumsformaten und Ergänzung fehlender Werte, zu Verzögerungen, da wiederholt Optimierungen notwendig waren. Diese Bereiche erforderten eine intensivere Entwicklung und Testphase, was zu geringfügigen Verzögerungen in der Zeitplanung führte.

Phase	Geplant	Tatsächlich	Differenz
Analyse	2 h	2 h	
Entwurf	4,5 h	3 h	-1,5 h
Implementierung und Testen	34 h	37 h	+3 h
Deployment	1 h	1 h	
Dokumentation	34 h	36 h	+2 h
Gesamt	**75,5 h**	**79 h**	

Tabelle 3: Zeitaufwand Soll-Ist-Abgleich

8.3 Lessons Learned

Während der Projektumsetzung konnte ich wertvolle Erfahrungen in der Automatisierung von Datenprozessen und der Erstellung benutzerfreundlicher GUIs sammeln. Besonders anspruchsvoll gestaltete sich die Datenbereinigung, da die Integration von Mechanismen zur Erkennung und Korrektur fehlerhafter Daten viele Testläufe und Anpassungen erforderte. Eine

wichtige Erkenntnis war, dass eine gut durchdachte Planung der Bereinigungslogiken notwendig ist, um flexible und skalierbare Lösungen zu entwickeln.

Die Implementierung der Funktion **save_charts()** zur Erstellung des Excel-Berichts und der Visualisierung zeigte mir die Bedeutung von detaillierten Tests und iterativer Optimierung. Die enge Integration von Matplotlib und OpenPyXL zur Visualisierung und Speicherung der Daten war komplexer als erwartet, aber notwendig, um qualitativ hochwertige Reports zu erstellen.

Eine weitere wichtige Lektion war die Notwendigkeit einer umfassenden Teststrategie, die frühzeitig und regelmäßig während der Implementierung eingesetzt wurde. Dies hat sich als entscheidend erwiesen, um die Funktionalität des Tools sicherzustellen und Fehler frühzeitig zu erkennen.

Besonders begeistert hat mich die Programmierung mit Python. Schon während des IHK-Kurses habe ich gemerkt, dass ich eine Leidenschaft für diese Programmiersprache habe. Es hat mir viel Spaß gemacht, den Code zu entwickeln und zu sehen, wie das Projekt schrittweise Form annahm. Ich bin stolz darauf, selbstständig einen Code entwickelt zu haben, der nicht nur funktional ist, sondern auch einen echten Mehrwert bieten könnte, würde man diesen in einem operativen betrieblichen Umfeld implementieren.

8.4 Fazit und Ausblick

Das Projekt demonstriert, wie durch Automatisierung zeitintensive manuelle Prozesse optimiert und die Datenqualität erheblich gesteigert werden können. Das Data Cleaning Tool hat bewiesen, dass es eine effiziente Lösung bietet, um Rohdaten zu bereinigen und in verwertbare Reports zu überführen.

Ein wesentlicher Aspekt für die Zukunft ist die Möglichkeit, das Tool für weitere Reporting-Arten zu erweitern, wie z.B. Vertriebsreportings für andere Geschäftsbereiche oder zusätzliche Unternehmenskennzahlen wie Liefertermintreue und Qualitätsreports. Diese Erweiterungen würden es ermöglichen, das Tool flexibler für verschiedene Datenanforderungen im Unternehmen einzusetzen.

Zusammenfassend lässt sich sagen, dass das Tool über die aktuellen Funktionen hinaus viel Potenzial bietet, um weitere Geschäftsanforderungen abzudecken und sich durch technologische Weiterentwicklungen nahtlos in die IT-Landschaft eines Unternehmens einzufügen.

Literaturverzeichnis

Dr. Krypczyk, Veikko u. Bochkor, Elena, 2022. *Handbuch für Softwareentwickler*, 2. überarbeitete und aktualisierte Auflage, Rheinwerk Verlag

Theis, Thomas, 2024. *Einstieg in Python - Ideal für Programmiereinsteiger*, 8. aktualisierte Auflage , Rheinwerk Verlag

Reichel, Erwin, 2024. *Lehrmaterialien, Unterrichtsinhalte und bereitgestellte Dokumentationen zum Thema Python-Programmierung, im Rahmen des Zertifikationslehrgangs Software Developer (IHK)*, Kurs xxxx, didaris

Mitsch, Mike, 2024. *Lehrmaterialien, Unterrichtsinhalte und bereitgestellte Dokumentationen zum Thema Flussdiagramme, Programmablaufplan, Projektmanagement, Wasserfall-Model, im Rahmen des Zertifikationslehrgangs Software Developer (IHK)*, Kurs xxxx, didaris

YouTube-Kanal: The Morpheus Tutorials - *https://www.youtube.com/@TheMorpheusTutorials Data Science in Python, Pandas Tutorial Serie*

Stack Overflow Forum, 2024. *https://stackoverflow.com/ Beiträge und Diskussionen zur Behandlung von Datumsformaten und Datenbereinigung in Pandas*

freeCodeCamp Tutorials, 2024. *https://www.freecodecamp.org/ Data Visualization with Matplotlib – a Step by Step Guide*

Datacamp Tutorials, 2024. *https://www.datacamp.com/ Datenbearbeitung mit Pandas*

OpenAI, 2024. ChatGPT – *https://chatgpt.com/ Unterstützung bei der Implementierung und Optimierung von Funktionen wie save_charts() und clean_data() für das Projekt*

Anhang

A1 Amortisationsberechnung

Amortisationsbrechnung *	Manuell	Automatisiert
Anfallende Tätigkeiten	Minuten	Minuten
Rohdaten extrahieren	10	10
Duplikate überprüfen	8	0,2
Datumsformate überprüfen	5	0,2
Verkäufernamen überprüfen	6	0,2
Region überprüfen	5	0,2
Gesamtsumme berechnen	4	0,2
Reporting Tabelle erstellen	7	0,2
Umsatzdiagramme erstellen	6	0,2
Bearbeitungszeit gesamt in min	51	11,4
Anzahl Vorgänge pro Jahr	12	12
min pro Jahr für alle Vorgänge	612	136,8
Stunden pro Jahr	10,2	2
Kosten gesamt pro Jahr **	408 €	91 €

* Die Berechnung basiert auf Schätzungen und Annahmen. Die Berechnung ist stark vereinfacht.
** Es wird ein Stundensatz von 40 Euro einschl. Personalnebenkosten zugrunde gelegt.

Jährliche Einsparung:	317 €
Amortisationsdauer:	4.368 € / 317 €
	≈ 13,78 Jahre
	≈ 165 Monate

Tabelle 4: Amortisationsberechnung

A2 Ist-Prozess

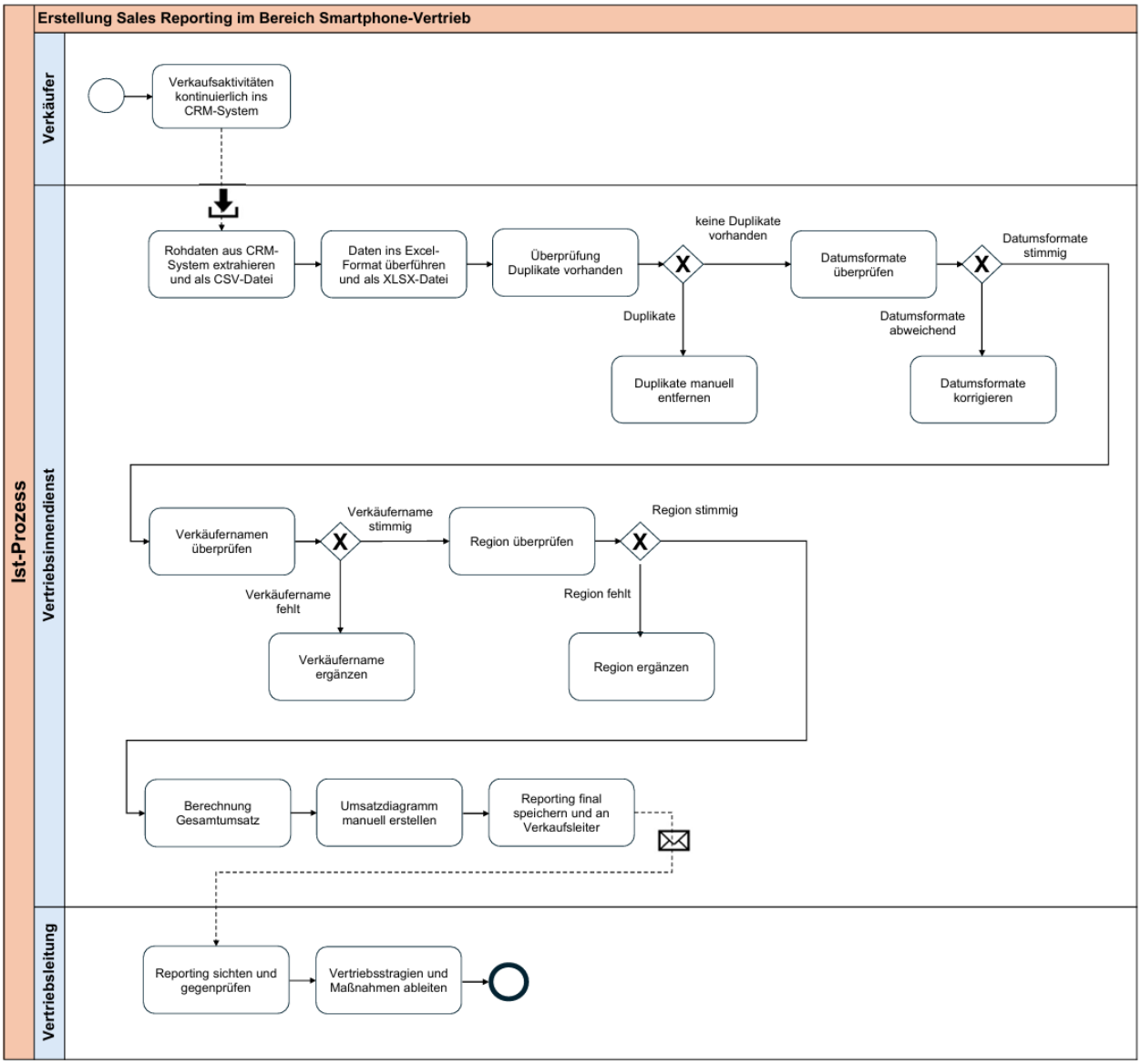

Abbildung 7: Ist-Prozess zur Erstellung des Sales Reportings

A3 Soll-Prozess

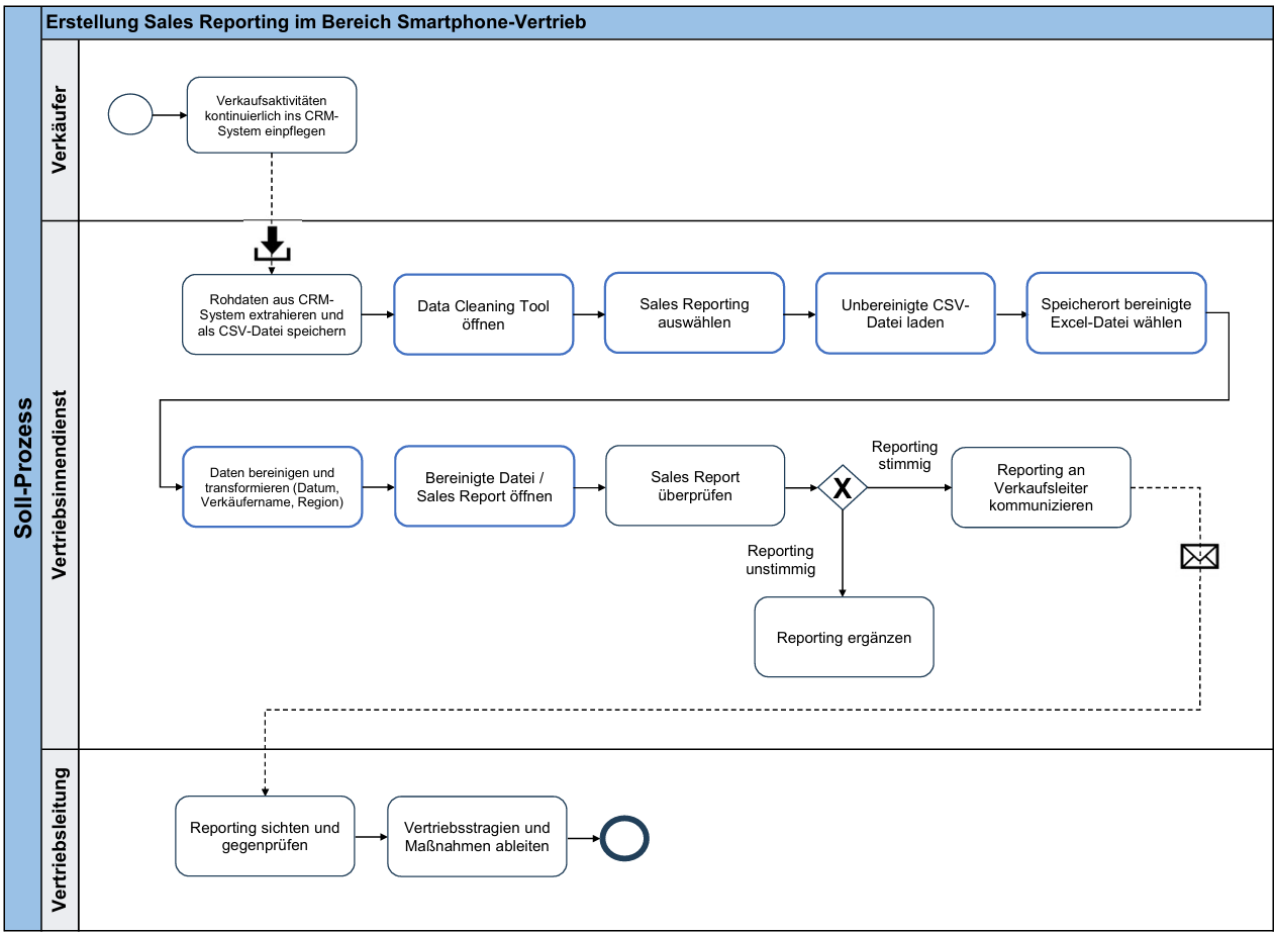

Abbildung 8: Soll-Prozess zur Erstellung des Sales Reportings

A4 Use-Case Diagramm

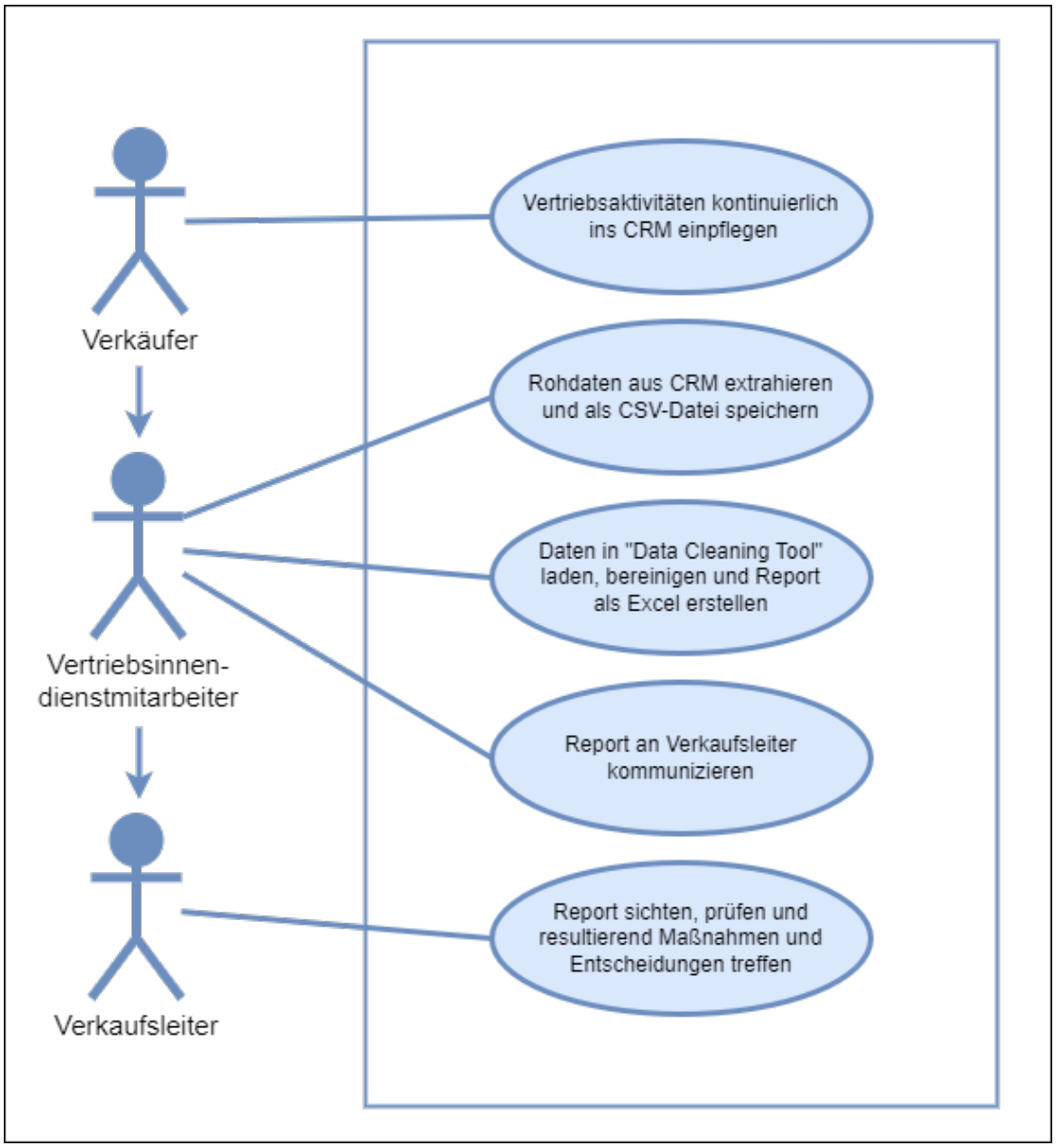

Abbildung 9: Use-Case-Diagramm

A5 Programmablaufplan

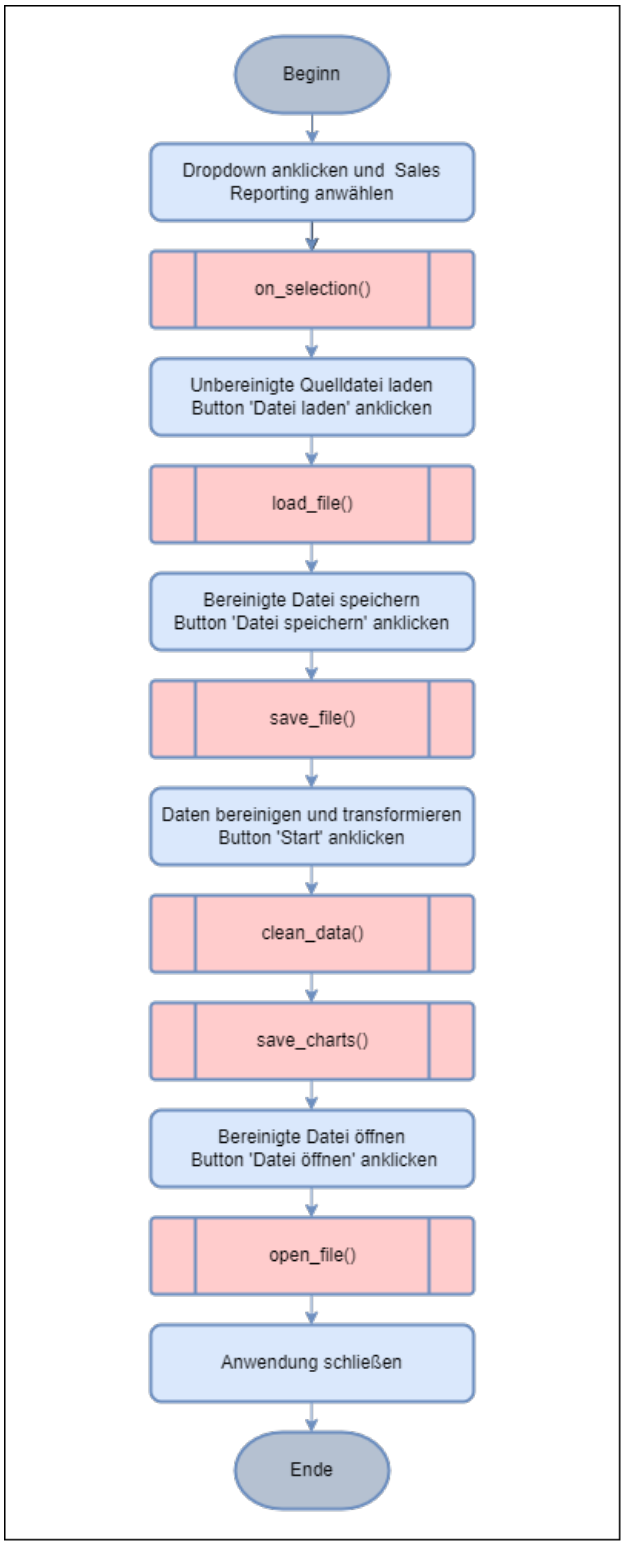

Abbildung 10: Programmablaufplan

A6 Struktogramm

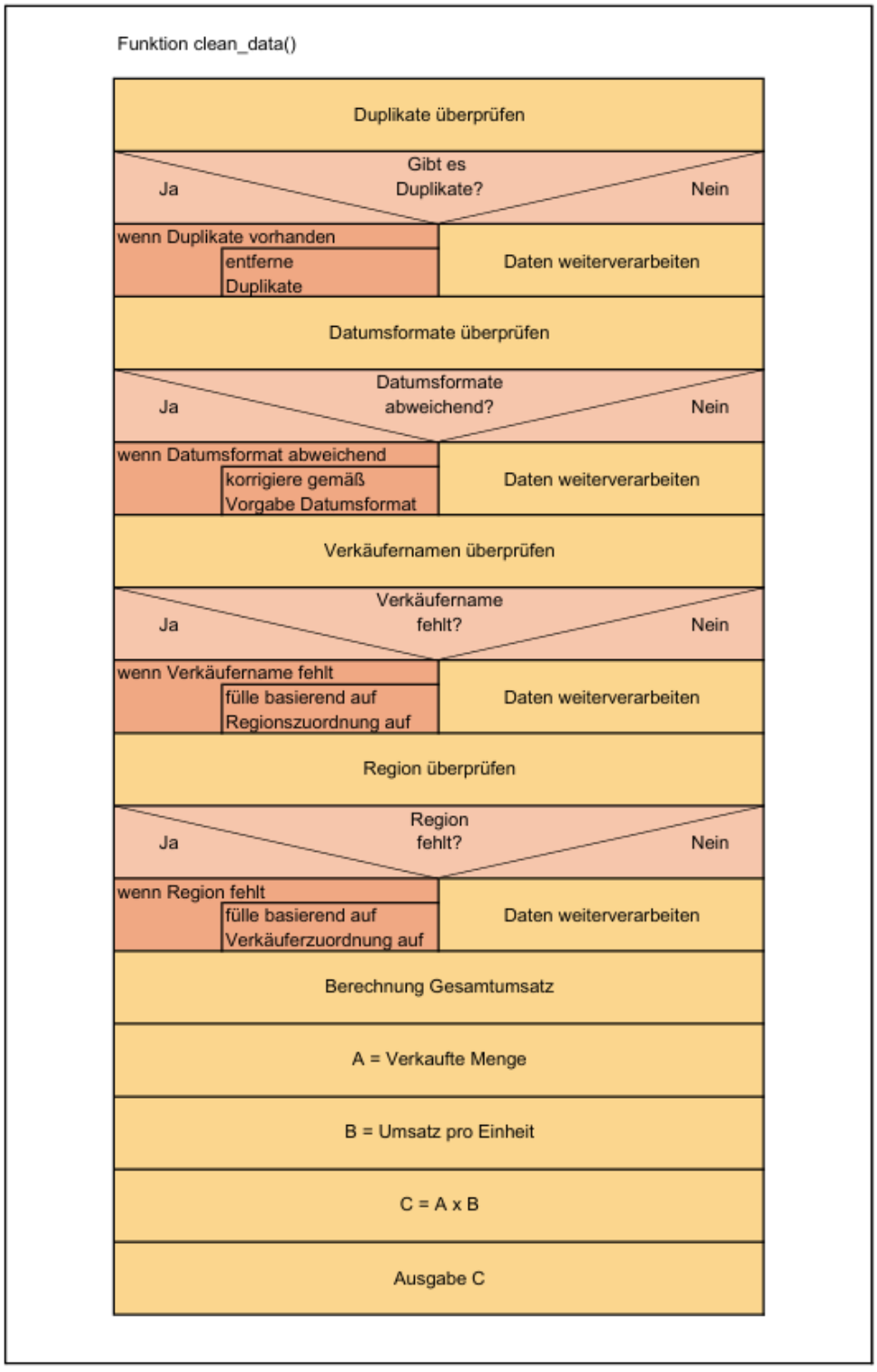

Abbildung 11: Struktogramm der Funktion clean_data()

A7 Benutzerdokumentation

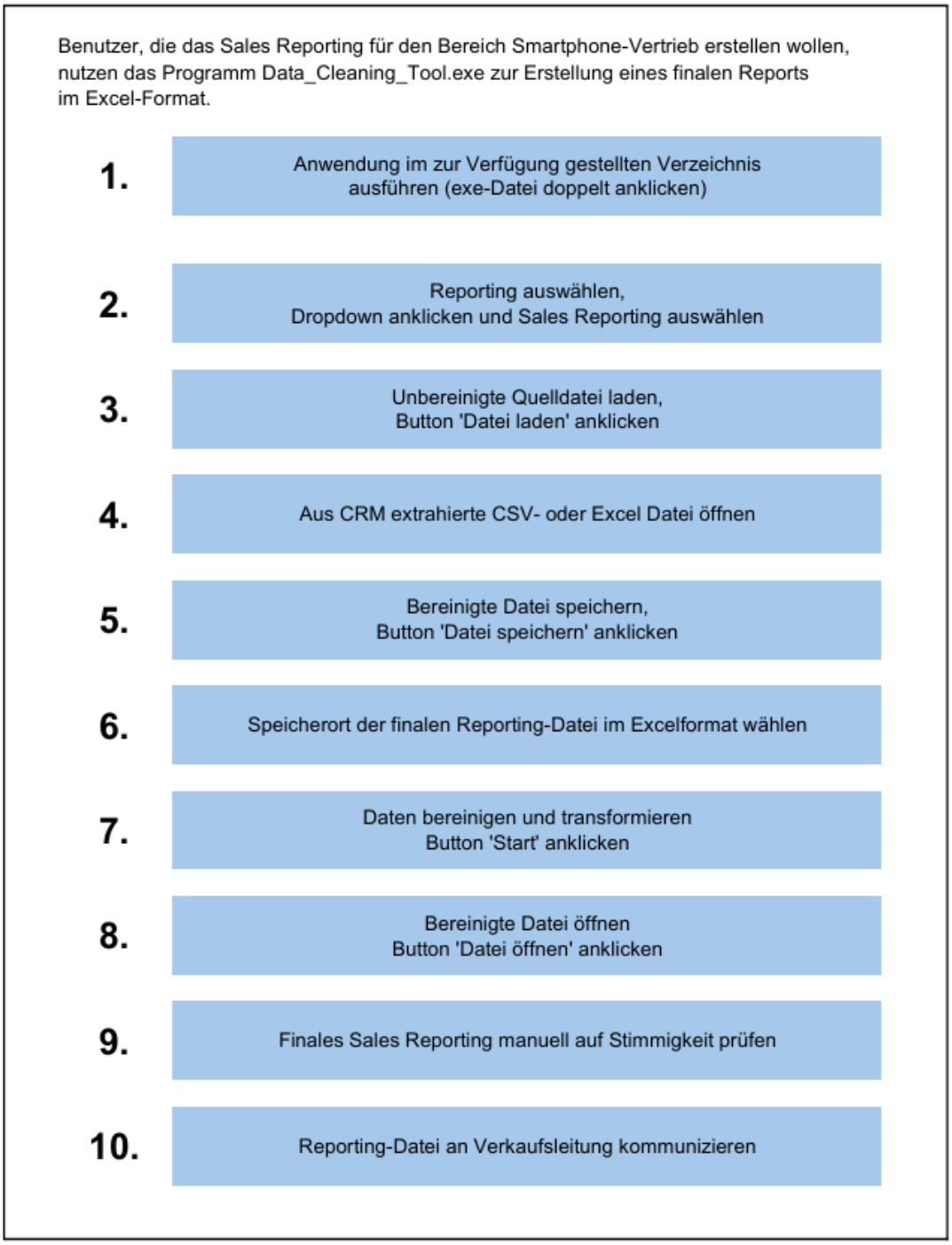

Abbildung 12: Benutzerdokumentation (Hinweise für Anwender*innen)

A8 Quellcode

```python
############################################################
# Entwickler: Christopher Haase                            #
# Kurs: Software Developer (IHK) [redacted]                #
# Erstellungsdatum: 05.09.2024                             #
# Letzte Änderung: 23.10.2024                              #
# Version: 1.0                                             #
# -------------------------------------------------------- #
# Projektarbeit: Data Cleaning Tool                        #
# Beschreibung: Entwicklung einer GUI basierten Daten-     #
# bereinigungs- und Transformationspipeline in Python      #
# -------------------------------------------------------- #
# Kontakt: [redacted]                                      #
############################################################

import customtkinter as ctk
from tkinter import filedialog, messagebox
import pandas as pd
import os
import subprocess
import webbrowser
import matplotlib.pyplot as plt
from openpyxl.drawing.image import Image
import matplotlib.ticker as ticker
import tempfile
import locale

# Konfiguration als Konstanten
WINDOW_TITLE = "Data Cleaning Tool"
WINDOW_SIZE = "520x670"
LABEL_FONT = "Helvetica Neue"
BUTTON_FONT = ("Helvetica Neue", 12, "bold")
BUTTON_COLOR = "#007AFF"
PROGRESS_BAR_WIDTH = 415


class DataCleaningApp:
    def __init__(self, root=None):
        if root is not None:
            self.root = root
            self._setup_window()
            self._create_ui()
        self.data = None
        self.cleaned_data = None  # Neue Variable für bereinigte Daten
        self.last_saved_path = None
        self.progress_value = 0

    def _setup_window(self):
        # Initialisierung Haupteigenschaften des Fensters
        if self.root:
            self.root.title(WINDOW_TITLE)
            self.root.geometry(WINDOW_SIZE)
            self.root.resizable(False, False)

    def _create_ui(self):
        # Erstellung gesamte Benutzeroberfläche
        if self.root:
            self._create_selection_menu()
            self.header_label = self._create_label("")
            self.load_button = self._create_section(
                "1. Unbereinigte Quelldatei laden:", "Datei laden", self.load_file
            )
            self.save_button = self._create_section(
                "2. Bereinigte Datei speichern:", "Datei speichern", self.save_file
            )
            self.process_button = self._create_section(
                "3. Daten bereinigen und transformieren:", "Start", self.process_data
            )
            self.open_button = self._create_section(
                "4. Bereinigte Datei öffnen:", "Datei öffnen", self.open_file
            )
            self._create_progress_bar()
            self._create_footer_buttons()

    def _create_selection_menu(self):
        # Erstellt das Auswahlmenü für die Reporting-Optionen.
        self.selection_frame = ctk.CTkFrame(self.root, corner_radius=12, fg_color="#F2F2F7", border_width=1,
                                            border_color="#D1D1D6")
        self.selection_frame.pack(pady=10, padx=20, fill="x")

        selection_label = self._create_label(
            "Reporting auswählen:", parent=self.selection_frame, font_size=14, bold=True
        )
        selection_label.pack(pady=5)

        self.selection_var = ctk.StringVar(value="")
        options = ["", "Sales Reporting"]

        selection_menu = ctk.CTkOptionMenu(
            self.selection_frame, values=options, command=self._on_selection, variable=self.selection_var,
            button_color=BUTTON_COLOR, button_hover_color="#0056D2", fg_color="#FFFFFF",
            dropdown_text_color="black", dropdown_fg_color="#F9F9F9", dropdown_hover_color="#E5E5EA",
            text_color="black"
        )
        selection_menu.pack(pady=5)
```

Anhang

```
95
96      def _create_label(self, text, parent=None, font_size=13, bold=False):
97          # Hilfsfunktion zum Erstellen von Labels
98          if parent is None:
99              parent = self.root
100         font_style = (LABEL_FONT, font_size, "bold" if bold else "normal")
101         return ctk.CTkLabel(
102             parent, text=text, wraplength=450, justify="center", font=font_style, text_color="#3A3A3C"
103         )
104
105     def _create_section(self, description_text, button_text, command):
106         # Erstellung Abschnitt mit einer Beschreibung und einem Button
107         frame = ctk.CTkFrame(self.root, corner_radius=12, fg_color="#F2F2F7", border_width=1, border_color="#D1D1D6")
108         frame.pack(pady=10, padx=20, fill="x")
109         label = self._create_label(description_text, parent=frame)
110         label.pack(pady=5)
111         button = ctk.CTkButton(
112             frame, text=button_text, command=command, corner_radius=8,
113             font=BUTTON_FONT, hover_color="#0056D2",
114             fg_color=BUTTON_COLOR, text_color="white"
115         )
116         button.pack(pady=5)
117         return button
118
119     def _create_progress_bar(self):
120         # Erstellung Fortschrittsanzeige und zugehörige Label
121         progress_frame = ctk.CTkFrame(self.root, fg_color="#F2F2F7")
122         progress_frame.pack(pady=10, padx=20, fill="x")
123
124         self.progress_bar = ctk.CTkProgressBar(
125             progress_frame, width=PROGRESS_BAR_WIDTH, height=20, progress_color=BUTTON_COLOR
126         )
127         self.progress_bar.set(0)
128         self.progress_bar.pack(side="left", pady=10, padx=(20, 2))
129
130         self.progress_label = self._create_label("0%", parent=progress_frame, font_size=10, bold=True)
131         self.progress_label.pack(side="right", padx=(0, 10))
132
133     def _create_footer_buttons(self):
134         # Erstellung Buttons Fußbereich
135         help_button = ctk.CTkButton(
136             self.root, text="Hilfe", command=self._open_help_email, corner_radius=8,
137             font=("Helvetica Neue", 10, "bold"), fg_color="#F2F2F7", text_color=BUTTON_COLOR,
138             hover_color="#E5E5EA", width=50
139         )
140         help_button.pack(side="right", anchor="se", padx=20, pady=10)
141
142         explanation_button = ctk.CTkButton(
143             self.root, text="Erklärung", command=self._show_welcome_message, corner_radius=8,
144             font=("Helvetica Neue", 10, "bold"), fg_color="#F2F2F7", text_color=BUTTON_COLOR,
145             hover_color="#E5E5EA", width=70
146         )
147         explanation_button.pack(side="left", anchor="sw", padx=20, pady=10)
148
149     def update_progress(self, value):
150         # Aktualisierung Fortschrittsanzeige
151         self.progress_bar.set(value)
152
153         progress_texts = {
154             0.0: "0%",
155             0.33: "33%",
156             0.66: "66%",
157             1.0: "100%"
158         }
159         self.progress_label.configure(text=progress_texts.get(value, ""))
160
161     def load_file(self):
162         # Laden einer CSV- oder Excel-Datei
163         if not self.selection_var.get():
164             messagebox.showerror("Fehler", "Bitte Reporting auswählen.")
165             return
166         file_path = filedialog.askopenfilename(
167             filetypes=[("CSV Dateien", "*.csv"), ("Excel Dateien", "*.xlsx")]
168         )
169         if file_path:
170             self.data = pd.read_csv(file_path) if file_path.endswith(".csv") else pd.read_excel(file_path)
171             messagebox.showinfo("Erfolg", "Datei erfolgreich geladen.")
172             self.update_progress(0.33)
173             self.load_button.configure(text="Erfolgreich erledigt", fg_color="green", state="disabled")
174
175     def save_file(self):
176         # Speichern bereinigte Daten in Excel-Datei
177         if self.data is None:
178             messagebox.showerror("Fehler", "Keine Datei geladen.")
179             return
180         save_path = filedialog.asksaveasfilename(
181             defaultextension=".xlsx",
182             filetypes=[("Excel Dateien", "*.xlsx"), ("Alle Dateien", "*.*")]
183         )
184         if save_path:
185             self.cleaned_data = self.clean_data(self.data)
186             self.cleaned_data.to_excel(save_path, index=False)
187             self.last_saved_path = save_path
188             messagebox.showinfo("Erfolg", "Datei erfolgreich gespeichert.")
189             self.update_progress(0.66)
190             self.save_button.configure(text="Erfolgreich erledigt", fg_color="green", state="disabled")
```

Anhang

```
191
192     @staticmethod
193     def _adjust_column_widths(workbook):
194         # Anpassung Spaltenbreiten an Inhalt an
195         for worksheet in workbook.worksheets:
196             for column_cells in worksheet.columns:
197                 max_length = max(len(str(cell.value)) for cell in column_cells)
198                 adjusted_width = (max_length + 2)
199                 worksheet.column_dimensions[column_cells[0].column_letter].width = adjusted_width
200
201     def process_data(self):
202         # Bereinigung Daten und Erstellung Diagramme
203         if self.cleaned_data is None:
204             messagebox.showerror("Fehler", "Keine bereinigten Daten vorhanden. Bitte Schritt 2 ausführen.")
205             return
206         self.save_charts(self.cleaned_data)
207         self.update_progress(1.0)
208         self.process_button.configure(text="Erfolgreich erledigt", fg_color="green", state="disabled")
209
210     @staticmethod
211     def clean_data(df):
212         # Bereinigung Datensatz und Korrektur Datumsformate
213
214         # Entfernen von doppelten Einträgen, außer dem Kommentar
215         df = df.drop_duplicates(
216             subset=['Datum', 'Verkäufer', 'Region', 'Produkt', 'Verkaufte Menge', 'Umsatz pro Einheit', 'Gesamtumsatz'],
217             keep='first'
218         ).copy()
219
220         # Bereinigung Datum und Konvertierung
221         def parse_date(date):
222             try:
223                 return pd.to_datetime(date, errors='coerce').strftime('%Y-%m-%d')
224             except ValueError:
225                 return None
226
227         # Direkte Zuweisung zu einer Spalte
228         df.loc[:, 'Datum'] = df['Datum'].apply(parse_date)
229
230         # Ausfüllen fehlender Datumsangaben mit nächst gültigem Datum
231         df.loc[:, 'Datum'] = df['Datum'].ffill()
232
233         # Ausfüllen fehlender Verkäufernamen basierend auf Region
234         def fill_verkaeufer(row):
235             if pd.isna(row['Verkäufer']):
236                 if row['Region'] == 'AMERICAS':
237                     return 'Peter Schmidt'
238                 elif row['Region'] == 'EMEA':
239                     return 'Stefan Berger'
240                 elif row['Region'] == 'ASIA':
241                     return 'Dirk Donner'
242             return row['Verkäufer']
243
244         df.loc[:, 'Verkäufer'] = df.apply(fill_verkaeufer, axis=1)
245
246         # Einheitliche Schreibweise der Verkäufernamen
247         df.loc[:, 'Verkäufer'] = df['Verkäufer'].str.title()
248
249         # Ausfüllen fehlender Regionsnamen basierend auf Verkäufer
250         def fill_region(row):
251             if pd.isna(row['Region']):
252                 if row['Verkäufer'] == 'Peter Schmidt':
253                     return 'AMERICAS'
254                 elif row['Verkäufer'] == 'Stefan Berger':
255                     return 'EMEA'
256                 elif row['Verkäufer'] == 'Dirk Donner':
257                     return 'ASIA'
258             return row['Region']
259
260         df.loc[:, 'Region'] = df.apply(fill_region, axis=1)
261
262         # Berechnung Monat und Gesamtumsatz
263         df.loc[:, 'Monat'] = pd.to_datetime(df['Datum']).dt.to_period('M')
264         df.loc[:, 'Gesamtumsatz'] = df['Verkaufte Menge'] * df['Umsatz pro Einheit']
265
266         return df
267
268     def save_charts(self, df):
269         # Erstellung Balkendiagramme und Speicherung in Excel-Datei
270         # Darstellung Monate auf Deutsch
271         locale.setlocale(locale.LC_TIME, 'de_DE.UTF-8')
272
273         df['Monat'] = pd.to_datetime(df['Datum']).dt.to_period('M')
274         df['Gesamtumsatz'] = df['Verkaufte Menge'] * df['Umsatz pro Einheit']
275
276         monthly_sales = df.groupby('Monat')['Gesamtumsatz'].sum()
277
278         # Erstellung erstes Diagramm: Gesamtumsatz pro Monat
279         fig, ax = plt.subplots(figsize=(10, 6))
280         monthly_sales.plot(kind='bar', ax=ax)
281         ax.set_xticklabels([period.strftime('%B %Y') for period in monthly_sales.index], rotation=45, ha="right")
282         ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'{x:,.0f} €'))
283         ax.set_title("Gesamtumsatz pro Monat")
284         ax.set_xlabel("Monat")
285         ax.set_ylabel("Gesamtumsatz in €")
286         ax.grid(True)
```

Anhang

```
287
288         # Speicherung des ersten Diagramms
289         temp_file_1 = os.path.join(tempfile.gettempdir(), "chart1.png")
290         plt.tight_layout()
291         plt.savefig(temp_file_1)
292         plt.close()
293
294         # Erstellung zweites Diagramm: Umsatz pro Verkäufer pro Monat
295         seller_monthly_sales = df.groupby(['Monat', 'Verkäufer'])['Gesamtumsatz'].sum().unstack()
296
297         fig2, ax2 = plt.subplots(figsize=(10, 6))
298         seller_monthly_sales.plot(kind='bar', stacked=True, ax=ax2)
299         ax2.set_xticklabels([period.strftime('%B %Y') for period in seller_monthly_sales.index], rotation=45,
300                             ha="right")
301         ax2.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'{x:,.0f} €'))
302         ax2.set_title("Umsatz pro Verkäufer pro Monat")
303         ax2.set_xlabel("Monat")
304         ax2.set_ylabel("Umsatz in €")
305         ax2.grid(True)
306
307         # Speicherung des zweiten Diagramms
308         temp_file_2 = os.path.join(tempfile.gettempdir(), "chart2.png")
309         plt.tight_layout()
310         plt.savefig(temp_file_2)
311         plt.close()
312
313         # Export aller Diagramme in Excel-Datei
314         if self.last_saved_path:
315             with pd.ExcelWriter(self.last_saved_path, engine='openpyxl') as writer:
316                 df.to_excel(writer, sheet_name='Bereinigte Daten', index=False)
317                 workbook = writer.book
318                 worksheet = workbook.create_sheet(title="Umsatzdiagramm")
319
320                 # Hinzufügen erstes Diagramm
321                 img1 = Image(temp_file_1)
322                 worksheet.add_image(img1, 'A1')
323
324                 # Hinzufügen zweites Diagramm
325                 img2 = Image(temp_file_2)
326                 worksheet.add_image(img2, 'A34')
327
328                 self._adjust_column_widths(workbook)
329                 workbook._sheets = [worksheet] + [workbook['Bereinigte Daten']]
330
331             # Löschung temporärer Dateien
332             os.remove(temp_file_1)
333             os.remove(temp_file_2)
334
335         messagebox.showinfo("Erfolg", "Daten erfolgreich bereinigt und Diagramme hinzugefügt.")
336
337     def open_file(self):
338         # Öffnen zuletzt gespeicherte Datei
339         if self.last_saved_path:
340             if os.name == 'nt':
341                 os.startfile(self.last_saved_path)
342             elif os.name == 'posix':
343                 subprocess.call(('open', self.last_saved_path))
344             else:
345                 messagebox.showerror("Fehler", "Das Betriebssystem wird nicht unterstützt.")
346         else:
347             messagebox.showerror("Fehler", "Es wurde noch keine Datei gespeichert.")
348
349     @staticmethod
350     def _open_help_email():
351         # Öffne Outlook mit Adresse
352         webbrowser.open("mailto:test@test.de")
353
354     @staticmethod
355     def _show_welcome_message():
356         # Anzeige Willkommensnachricht
357         messagebox.showinfo(
358             "Erklärung",
359             "Dieses Tool ermöglicht es Ihnen, CSV- oder Excel-Dateien zu laden, "
360             "zu bereinigen und die Ergebnisse zu speichern. Sie können die bereinigte Datei "
361             "anschließend öffnen."
362         )
```

```
364     def _on_selection(self, selected_option):
365         # Auswahl Dropdown Menü
366         if not selected_option:
367             self.header_label.pack_forget()
368             self.load_button.configure(
369                 text="Datei laden", fg_color=BUTTON_COLOR, state="normal",
370                 command=lambda: messagebox.showerror("Fehler", "Bitte Reporting auswählen.")
371             )
372             self.save_button.configure(
373                 text="Datei speichern", fg_color=BUTTON_COLOR, state="normal"
374             )
375             self.process_button.configure(
376                 text="Start", fg_color=BUTTON_COLOR, state="normal"
377             )
378             self.open_button.configure(
379                 text="Datei öffnen", fg_color=BUTTON_COLOR, state="normal"
380             )
381             self.update_progress(0.0)
382         else:
383             self.load_button.configure(command=self.load_file)
384             if selected_option == "Sales Reporting":
385                 self.header_label.configure(
386                     text="Sales Reporting:\n"
387                          "Analyse der monatlichen Gesamtumsätze und der Einzelumsätze pro Verkäufer"
388                          " im Bereich Smartphone-Vertrieb."
389                 )
390             self.header_label.pack(pady=8, after=self.selection_frame)
391
392
393 if __name__ == "__main__":
394     main_root = ctk.CTk()  # Erstellung Hauptfenster
395     main_root.iconbitmap('icon.ico')  # Icon für Fenster
396     app = DataCleaningApp(main_root)
397     main_root.mainloop()
398
```

Abbildung 13: Quellcode