

BACHELOR THESIS  
Framarz Alizadeh

# Automatisierte Rechnungs- und Dokumentenverarbeitung: Konzeption und Implementierung eines Chatbotbasierten Systems zur Erstellung und Ablage von Rechnungen mit ergänzender Web-Anwendung für Verwaltung und Zugriff.

---

FAKULTÄT INFORMATIK UND DIGITALE GESELLSCHAFT

Faculty of Computer Science and Digital Society

---

HOCHSCHULE FÜR ANGEWANDTE  
WISSENSCHAFTEN HAMBURG  
Hamburg University of Applied Sciences



Framarz Alizadeh

Automatisierte Rechnungs- und  
Dokumentenverarbeitung: Konzeption und  
Implementierung eines Chatbotbasierten Systems  
zur Erstellung und Ablage von Rechnungen mit  
ergänzender Web-Anwendung für Verwaltung und  
Zugriff.

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung  
im Studiengang *Bachelor of Science Informatik Technischer Systeme*  
der Fakultät Informatik und digitale Gesellschaft  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt  
Zweitgutachter: Prof. Dr. Ulrike Steffens

Eingereicht am: 07. Juni 1954



**Framarz Alizadeh**

## **Thema der Arbeit**

Automatisierte Rechnungs- und Dokumentenverarbeitung: Konzeption und Implementierung eines Chatbotbasierten Systems zur Erstellung und Ablage von Rechnungen mit ergänzender Web-Anwendung für Verwaltung und Zugriff.

## **Stichwörter**

Rechnungsautomatisierung, Dokumentenmanagement, Chatbot-Systeme, Cloud-Speicherung, Webanwendungen, Prozessdigitalisierung

## **Kurzzusammenfassung**

Die manuelle Erstellung, Verwaltung und Ablage von Rechnungen stellt in vielen kleinen und mittleren Unternehmen eine erhebliche organisatorische und zeitliche Belastung dar. Insbesondere fehleranfällige Arbeitsschritte, redundante Dateneingaben sowie unstrukturierte Ablageprozesse erschweren ein effizientes Dokumentenmanagement. Ziel dieser Arbeit ist die Konzeption und Implementierung eines automatisierten Systems, das den gesamten Rechnungsprozess digitalisiert und durch einen Chatbot unterstützt. Die Datenerfassung erfolgt über ein dialogbasiertes Chatbot-System, das mithilfe der Plattform Make.com implementiert wurde und Kundendaten sowie Rechnungsinformationen strukturiert in Airtable speichert. Auf Basis dieser Daten werden Rechnungsdokumente automatisiert in Google Docs generiert und revisionssicher in einer hierarchischen Ordnerstruktur in Google Drive abgelegt. Ergänzend wurde eine moderne Webanwendung entwickelt, die als Verwaltungs- und Zugriffssystem dient und die Kundendaten sowie die generierten Dokumente übersichtlich darstellt. Die Web-App basiert auf React, TypeScript und Tailwind CSS und kommuniziert über serverlose Edge Functions mit der Airtable- und Google-Drive-API. Das Gesamtsystem zeigt, dass der Einsatz moderner Cloud-Technologien und automatisierter Workflows die Prozessqualität erhöht, Fehler reduziert und eine deutliche Effizienzsteigerung ermöglicht. Die Arbeit demonstriert das Potenzial kombinierter Chatbot- und Web-Technologien für eine zukunftsorientierte Digitalisierung administrativer Geschäftsprozesse.

**Framarz Alizadeh**

---

## **Title of Thesis**

Automated invoice and document processing: Design and implementation of a chatbot-based system for creating and storing invoices with a supplementary web application for management and access.

## **Keywords**

Invoice Automation, Document Management, Chatbot Systems, Cloud Storage, Web Applications, Process Digitalization

## **Abstract**

The manual creation, management, and storage of invoices represents a significant organizational and time-consuming challenge for many small and medium-sized enterprises. Error-prone workflows, repeated data entry, and unstructured document storage limit the efficiency of traditional invoice processes. This thesis aims to design and implement an automated system that digitizes the entire invoice workflow and supports it through a chatbot-based interaction model. Data collection is performed via a dialog-driven chatbot built using Make.com, which stores customer and invoice information in Airtable in a structured manner. Based on this data, invoice documents are automatically generated using Google Docs and stored in a revision-proof, hierarchical folder structure in Google Drive. In addition, a modern web application was developed to provide administrative access and management features for customers and documents. The web application is built with React, TypeScript, and Tailwind CSS and communicates with Airtable and Google Drive through serverless edge functions. The system demonstrates that the use of modern cloud technologies and automated workflows can significantly improve process quality, reduce errors, and enhance operational efficiency. This thesis highlights the potential of combining chatbot technology and web-based interfaces to support the digital transformation of administrative business processes.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>ix</b>
<b>Tabellenverzeichnis</b>	<b>xi</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Motivation und Zielsetzung . . . . .	1
1.3 Forschungsfragen . . . . .	1
1.4 Vorgehensweise und Aufbau der Arbeit . . . . .	1
<b>2 Theoretische Grundlagen</b>	<b>2</b>
2.1 Digitalisierung administrativer Prozesse und Rechnungswesen . . . . .	2
2.2 Chatbots und dialogbasierte Systeme . . . . .	2
2.3 Cloudbasierte Datenhaltung und Integrationen . . . . .	2
2.4 Dokumentengenerierung in der Cloud . . . . .	2
2.5 Grundlagen moderner Webanwendungen . . . . .	2
<b>3 Technologischer Hintergrund</b>	<b>3</b>
3.1 Make.com als Integrations- und Automatisierungsplattform . . . . .	3
3.2 Airtable als Datenhaltung . . . . .	3
3.3 Google Docs und Google Drive . . . . .	3
3.4 Lovable Cloud / Supabase Edge Functions . . . . .	3
3.5 Frontend-Stack (React, TypeScript, Tailwind, shadcn/ui) . . . . .	3
<b>4 Konzeption des Systems</b>	<b>4</b>
4.1 Anforderungen und Use Cases . . . . .	4
4.2 Zielarchitektur und Systemübersicht . . . . .	7
4.3 Datenmodell und Datenflüsse . . . . .	9
4.4 Prozessablauf vom Chatbot bis zur Ablage . . . . .	12
4.4.1 Ablauf „Rechnung neu erstellen“ . . . . .	12

4.4.2	Ablauf „Dokument ablegen“ . . . . .	15
4.5	Sicherheits- und Datenschutzkonzept . . . . .	16
<b>5</b>	<b>Implementierung</b>	<b>17</b>
5.1	Implementierung des Chatbots (make.com-Szenarien) . . . . .	17
5.1.1	Implementierung des Chatbots (Make.com-Szenarien) . . . . .	17
5.2	Datenhaltung und Strukturen in Airtable . . . . .	20
5.3	Automatische Dokumentenerstellung in Google Docs . . . . .	20
5.4	Ablagestrategie in Google Drive (Jahr/Quartal/Monat) . . . . .	20
5.5	Implementierung der ClientHub WebApp . . . . .	20
5.5.1	Frontend (React, UI-Konzept) . . . . .	20
5.5.2	Edge Functions in Lovable Cloud . . . . .	20
5.5.3	API-Anbindung an Airtable und Google Drive . . . . .	20
<b>6</b>	<b>Evaluation</b>	<b>21</b>
6.1	Testkonzept und Testumgebung . . . . .	21
6.2	Funktionale Tests . . . . .	21
6.3	Performance-Analyse . . . . .	21
6.4	Vergleich: manueller vs. automatisierter Prozess . . . . .	21
6.5	Usability-Bewertung . . . . .	21
6.6	Grenzen des Systems und Risiken . . . . .	21
<b>7</b>	<b>Fazit und Ausblick</b>	<b>22</b>
7.1	Zusammenfassung der Ergebnisse . . . . .	22
7.2	Beantwortung der Forschungsfragen . . . . .	22
7.3	Beitrag der Arbeit und praktische Implikationen . . . . .	22
7.4	Ausblick und Weiterentwicklung . . . . .	22
	<b>Literaturverzeichnis</b>	<b>23</b>
<b>A</b>	<b>Anhang</b>	<b>25</b>
A.1	Verwendete Hilfsmittel . . . . .	25
	<b>Selbstständigkeitserklärung</b>	<b>26</b>



# Abbildungsverzeichnis



# Tabellenverzeichnis

A.1	Verwendete Hilfsmittel und Werkzeuge . . . . .	25
-----	--	----













# 1 Einleitung

## 1.1 Problemstellung

## 1.2 Motivation und Zielsetzung

## 1.3 Forschungsfragen

## 1.4 Vorgehensweise und Aufbau der Arbeit

## 2 Theoretische Grundlagen

2.1 Digitalisierung administrativer Prozesse und Rechnungswesen

2.2 Chatbots und dialogbasierte Systeme

2.3 Cloubasierte Datenhaltung und Integrationen

2.4 Dokumentengenerierung in der Cloud

2.5 Grundlagen moderner Webanwendungen

## 3 Technologischer Hintergrund

3.1 Make.com als Integrations- und Automatisierungsplattform

3.2 Airtable als Datenhaltung

3.3 Google Docs und Google Drive

3.4 Lovable Cloud / Supabase Edge Functions

3.5 Frontend-Stack (React, TypeScript, Tailwind, shadcn/ui)

## 4 Konzeption des Systems

### 4.1 Anforderungen und Use Cases

Die Situation und das Problem, das damit verbunden ist, sind wie folgt:

Das geplante System richtet sich vor allem an kleine Dienstleistungsfirmen, Einzelunternehmer sowie kleine und mittlere Gewerbebetriebe. Diese müssen regelmäßig Rechnungen erstellen und verwalten, verfügen jedoch weder über die Ressourcen noch über den Bedarf für umfassende Buchhaltungs- oder ERP-Systeme. Häufig mangelt es diesen Personen an IT-Kenntnissen und sie haben wenig Zeit. Deshalb benötigen sie einfache, zuverlässige und größtenteils automatisierte Prozesse. Auch Steuerberaterinnen und Steuerberater sind auf vollständig und strukturiert archivierte Rechnungsunterlagen angewiesen. Die Erkenntnisse aus Studien und Praxisberichten zeigen, dass in kleinen Unternehmen manuelle Rechnungsprozesse besonders zeitaufwendig und fehleranfällig sind. Außerdem resultieren sie häufig in Medienbrüchen und einem Mangel an Transparenz [2, 13, 6].

In der Praxis erfolgt die Erstellung von Rechnungen und Dokumenten in vielen dieser Unternehmen überwiegend manuell. Die Ablage geschieht häufig in unsystematischen Ordnerstrukturen. Dieser Ansatz ist zeitaufwendig und anfällig für Fehler, da Kundendaten und Rechnungspositionen wiederholt neu eingegeben werden müssen. Dadurch steigt das Risiko von Zahlendrehern, unvollständigen Daten oder fehlerhaften Summenberechnungen. Zudem erschwert eine unstrukturierte Ablage das spätere Wiederfinden einzelner Dokumente, insbesondere wenn Belege zu einem späteren Zeitpunkt für buchhalterische oder steuerliche Zwecke benötigt werden.

Rechnungen, die Fehler aufweisen oder verspätet erstellt werden, können den Zahlungseingang verzögern und damit die Liquidität der Unternehmen beeinträchtigen. Sind Dokumente unvollständig, unsortiert oder schwer auffindbar, verlängert sich nicht nur die Bearbeitungszeit von Kundenanfragen – was den professionellen Außenauftritt beeinträchtigen kann –, sondern auch der Aufwand, wenn Belege kurzfristig benötigt werden.

---

Die nachträgliche Aufbereitung und Abstimmung solcher Unterlagen kostet zusätzliche Zeit und Ressourcen und erhöht das Risiko, dass steuerliche Erklärungen fehlerhaft oder verspätet eingereicht werden.

#### Ziele und Anforderungen aus Sicht der Anwender

Die Entlastung im täglichen Verwaltungs- und Abrechnungsalltag steht aus Anwendersicht im Vordergrund. Das Ziel ist es, die Zeit und den Aufwand, die für die Erstellung und Verwaltung von Rechnungen benötigt werden, deutlich zu reduzieren, damit mehr Zeit für das eigentliche Kerngeschäft bleibt.

Gleichzeitig besteht das Bedürfnis nach einer übersichtlichen und verlässlichen Ablage für geschäftsrelevante Dokumente. Um Fehler zu vermeiden und die Effizienz der Arbeitsabläufe zu verbessern, ist es entscheidend, dass alle Daten vollständig, korrekt und einheitlich erfasst werden.

Außerdem ist die Zusammenarbeit mit externen Partnern, vor allem mit Steuerberaterinnen und Steuerberatern, von großer Bedeutung. Anwender gehen davon aus, dass alle benötigten Unterlagen aktuell, vollständig und gut strukturiert sind, um Rückfragen zu minimieren und zusätzlichen Abstimmungsaufwand zu vermeiden.

Diese Ziele resultieren in konkreten funktionalen Anforderungen an das System. Es muss den Nutzern ermöglichen, bestehende Kundendaten auszuwählen oder neue Kundendatensätze mit allen relevanten Stammdaten anzulegen. Darüber hinaus soll das System alle für die Rechnungserstellung erforderlichen Informationen erfassen. Dazu zählen unter anderem Leistungsbeschreibungen, Mengenangaben, Einzelpreise, Steuersätze und Zahlungsfristen, die strukturiert und nachvollziehbar eingegeben werden können.

Nach Abschluss der Datenerfassung stellt das System das erstellte Rechnungsdokument bereit und ermöglicht zudem die Ablage weiterer geschäftsrelevanter Dokumente, sodass diese später erneut abgerufen werden können.

Nicht nur die funktionalen, sondern auch die nicht-funktionalen Anforderungen sind von großer Bedeutung. Um auch Menschen ohne technische Vorkenntnisse zu erreichen, soll das System benutzerfreundlich gestaltet sein, sodass der Prozess der Rechnungserstellung und -verwaltung intuitiv durchlaufen werden kann. Eine positive Nutzungserfahrung setzt zudem kurze Reaktionszeiten voraus, um lange Wartezeiten für die Nutzer zu vermeiden.

Darüber hinaus ist es wichtig, dass das Erstellen neuer Rechnungen sowie das Ablegen von Dokumenten zügig erfolgt, damit erzeugte oder hochgeladene Dateien ohne spürbare Verzögerung zur Verfügung stehen. Das System muss außerdem zuverlässig und fehlertolerant arbeiten. Eingabefehler sollen durch geeignete Validierungen erkannt und abgefangen werden, während die Abläufe stabil und reproduzierbar ausgeführt werden.

Da personenbezogene Daten und Abrechnungsinformationen verarbeitet werden, kommt der Datensicherheit eine besondere Bedeutung zu. Zugangsdaten sind sicher zu speichern, die Kommunikation ist zu verschlüsseln und der Zugriff auf autorisierte Nutzer zu beschränken. Weiterhin soll die Systemarchitektur modular und erweiterbar gestaltet sein, um zukünftige Anforderungen oder zusätzliche Funktionen mit vertretbarem Aufwand integrieren zu können. Insgesamt orientiert sich das System damit an anerkannten Qualitätsmerkmalen nicht-funktionaler Anforderungen, wie sie in der Literatur beschrieben sind [12, 7].

#### Anwendungsfall „Rechnung neu erstellen“

Der Anwendungsfall „Rechnung neu erstellen“ beginnt, sobald der Nutzer die entsprechende Aktion auswählt und den Erstellungsprozess startet. Zunächst entscheidet der Nutzer, ob die Daten eines bestehenden Kunden verwendet oder ein neuer Kundendatensatz angelegt werden soll. Bei der Neuanlage werden alle erforderlichen Stammdaten wie Name oder Firmenbezeichnung, Adresse, Kontaktinformationen und gegebenenfalls steuerliche Angaben erfasst.

Im nächsten Schritt gibt der Nutzer die zu fakturierenden Leistungen an. Dazu zählen die Leistungsbeschreibung sowie die zugehörigen Mengen, Einzelpreise und gegebenenfalls anwendbare Steuersätze. Der Nutzer kann mehrere Rechnungspositionen nacheinander erfassen, sofern dies erforderlich ist.

Anschließend werden rechnungsbezogene Angaben wie Rechnungsnummer, Rechnungsdatum, Leistungszeitraum und Zahlungsfrist festgelegt. Nach Abschluss der Dateneingabe wird die Rechnung erstellt und dem Nutzer zur Verfügung gestellt.

Anwendungsfall „Dokument ablegen“ Der Anwendungsfall „Dokument ablegen“ beginnt, sobald der Nutzer die entsprechende Aktion auswählt, um geschäftsrelevante Dokumente im System zu archivieren. Hierzu lädt der Nutzer eine Datei, beispielsweise einen Beleg oder eine Rechnung, hoch.

---

Das System ordnet das Dokument dem entsprechenden Kunden zu und legt es strukturiert ab, sodass es später zuverlässig wiedergefunden werden kann. Nach dem Abschluss des Vorgangs steht das abgelegte Dokument dem Nutzer zur Einsicht und zum erneuten Abruf zur Verfügung.

## 4.2 Zielarchitektur und Systemübersicht

Die Zielarchitektur des Systems basiert auf mehreren klar abgegrenzten und lose gekoppelten Komponenten, die gemeinsam den End-to-End-Prozess der Rechnungs- und Dokumentenverarbeitung abbilden. Zu den zentralen Elementen zählen der WhatsApp-Chatbot zur Datenerfassung und Prozesssteuerung, Workflows auf der Automatisierungsplattform Make (make.com), eine Airtable-Datenbank als zentrale Datenquelle, Google Docs zur templatebasierten Dokumentenerstellung, Google Drive als Ablagesystem sowie die Web-Anwendung „ClientHub“ für Verwaltung und Zugriff. Die einzelnen Komponenten sind überwiegend über standardisierte Cloud-APIs miteinander verbunden. Dadurch können sie unabhängig voneinander betrieben und bei Bedarf flexibel erweitert werden.

Das zentrale Element des Systems ist der Chatbot, der über WhatsApp erreichbar ist. Er begleitet die Nutzer dialoggesteuert durch den Prozess der Rechnungsgenerierung, indem er entsprechende Anweisungen gibt. Dies umfasst die Auswahl eines bestehenden Kunden oder die Erstellung eines neuen Kundendatensatzes, die Festlegung der Rechnungspositionen sowie die Bestimmung von Rechnungsnummer, Leistungszeitraum, Zahlungsfrist und Zahlungsart. Ein weiteres Feature des Chatbots ist die Funktion „Dokument speichern“. Diese erlaubt es den Nutzern, Dateien wie Belege oder bereits erstellte Rechnungen hochzuladen, die anschließend im Kundenportal zur Verfügung gestellt werden. In beiden Szenarien leitet der Chatbot die erfassten Informationen in strukturierter Form an die nachgelagerten Arbeitsabläufe weiter. Er selbst führt keine direkten Abfragen an Datenbank- oder Speicherdiensten durch.

Eine Integrations- und Workflow-Plattform fungiert als Orchestrierungsschicht und implementiert die Logik der Automatisierung. Sie empfängt Nachrichten vom Chatbot, analysiert den aktuellen Dialogschritt und startet kontextabhängig unterschiedliche externe Dienste. Hierzu zählen insbesondere die Airtable-API zur Erstellung und Aktualisierung von Kunden-, Rechnungs- und Sitzungsdaten (z. B. des aktuellen Prozessschritts), die Google-Docs-API zum Öffnen und Befüllen eines vordefinierten Rechnungsmusters sowie die Google-Drive-API zur Erstellung von Ordnern und zum Hochladen von Dateien.

Ergänzend wird eine KI-Schnittstelle genutzt, um frei formulierte Texte oder Sprachnachrichten in strukturierte, deutschsprachige Rechnungspositionen zu überführen. Im Rahmen des Dialogs weist der Nutzer die Rechnungsnummer zu. Diese wird als Feld in Airtable gespeichert, um sicherzustellen, dass sie in allen nachfolgenden Prozessschritten konsistent verwendet wird.

Google Drive fungiert als unabhängiges Speichersystem, das insbesondere im Anwendungsfall „Dokument ablegen“ genutzt wird. Der Nutzer startet diesen Vorgang, indem er über den Chatbot eine Datei hochlädt, die durch einen Workflow in eine kundenbezogene Ordnerstruktur überführt wird. Die Verzeichnisse sind nach Jahr, Quartal und Monat gegliedert; ist ein Ordner noch nicht vorhanden, wird er dynamisch erstellt. Der archivierte Dateilink wird in Airtable gespeichert, sodass er für nachgelagerte Automatisierungen sowie für die Anzeige im Kundenportal verfügbar ist. Auf diese Weise entsteht eine konsistente und nachvollziehbare Archivierung, die sowohl interne Anforderungen als auch die Zusammenarbeit mit Steuerberaterinnen und Steuerberatern unterstützt.

Die Webanwendung „ClientHub“ ist eine browserbasierte React-Anwendung, die als eigenständiges Modul fungiert und ausschließlich serverseitige Edge-Funktionen nutzt, um auf externe Dienste zuzugreifen. Diese Edge-Funktionen bilden die Backend-Schicht der Architektur ab. Sie übernehmen die Interaktion mit der Airtable-API und der Google-Drive-API, bereiten die Daten für das Frontend auf und schützen gleichzeitig sensible Zugangsdaten vor einem direkten Zugriff aus dem Browser. Die Webanwendung stellt den Nutzern ein Dashboard mit aggregierten Metriken zur Verfügung. Darüber hinaus bietet sie eine Kundenliste mit Such- und Filteroptionen sowie eine Übersicht über die Ordnerstruktur des Kundenportals, über die archivierte Dokumente eingesehen und heruntergeladen werden können.

Das Design folgt bewusst den architektonischen Grundsätzen der losen Kopplung, der Modularität und der klaren Schichtentrennung. Die Frontend-Logik, die Orchestrierung der Integrationsprozesse und die Datenhaltung sind voneinander entkoppelt. Dadurch können Änderungen in einer Komponente, wie etwa der Austausch des Chatkanals oder die Erweiterung der Datenstruktur, mit nur minimalen Anpassungen in den übrigen Systemteilen umgesetzt werden. Durch den konsequenten Einsatz von Cloud-Diensten und standardisierten APIs ist keine eigene Serverinfrastruktur erforderlich. Betrieb, Skalierung und Verfügbarkeit werden größtenteils von den jeweiligen Plattformanbietern übernommen. Die Architektur weist damit Eigenschaften auf, die sie besonders geeignet für kleine Unternehmen machen, die mit begrenzten Ressourcen arbeiten, einen geringen



---

Wartungsaufwand benötigen und dennoch eine Lösung suchen, die flexibel mit wachsenden Anforderungen erweitert werden kann. Lose Kopplung, Modularität und eine klare Schichtentrennung gelten in der Softwarearchitektur als wesentliche Voraussetzungen für wartbare und erweiterbare Systeme. [1, 4, 5]. Der konsequente Einsatz wohldefinierter APIs wird als zentraler Baustein integrationsfähiger, verteilter Systeme beschrieben [14].

## 4.3 Datenmodell und Datenflüsse

Das zugrunde liegende Datenmodell und die zentralen Datenflüsse, die die dialogbasierte Rechnungserstellung sowie die Ablage geschäftsrelevanter Dokumente unterstützen, werden im folgenden Abschnitt erläutert. Die Daten werden in mehreren Airtable-Tabellen gespeichert, die gemeinsam den Chatbot-Dialog, die Rechnungserstellung und die Dokumentenablage in Google Drive abbilden. Die Tabellen dienen dabei nicht nur als persistenter Speicher fachlicher Informationen, sondern bilden zugleich die Grundlage für die Steuerung der Workflows in der Automatisierungsplattform, indem sie Zustandsinformationen und temporäre Eingaben strukturiert bereitstellen.

Im Datenmodell steht die Tabelle „User\_Sessions“ im Mittelpunkt. Sie fungiert als zentrale Instanz zur Verwaltung von Nutzerzuständen und zur Zugriffskontrolle innerhalb des Chatbot-Workflows. In dieser Tabelle werden die Telefonnummern der Nutzer gespeichert, sodass eingehende WhatsApp-Nachrichten validiert und ausschließlich registrierte Rufnummern zum Systemzugang zugelassen werden. Darüber hinaus enthält „User\_Sessions“ Zustandsvariablen wie „Current Step“ und „Next Step“, die zur Steuerung des dialogbasierten Ablaufs verwendet werden. Diese Informationen sind erforderlich, da die eingesetzte Automatisierungsplattform lediglich ein einzelnes Watch-Event für eingehende Nachrichten bereitstellt und keine native Unterstützung für eine zustandsabhängige Dialogführung bietet.

„User\_Sessions“ enthält ausschließlich dynamische, während des Gesprächsverlaufs veränderliche Felder. Diese dienen der temporären Speicherung von Nutzereingaben und ermöglichen eine iterative, dialogorientierte Datenerfassung. Ein zentrales Ziel dieser Struktur besteht darin, Korrekturen und Anpassungen durch die Nutzenden jederzeit zuzulassen. Gibt eine Person beispielsweise versehentlich ein falsches Rechnungsjahr an,

kann dieser Wert direkt im Chat korrigiert werden. Der entsprechende Eintrag in „User\_Sessions“ wird aktualisiert, und der Workflow setzt mit den korrigierten Daten fort, ohne dass der gesamte Dialog erneut durchlaufen werden muss.

Ein vergleichbares Vorgehen wird bei der Neuanlage von Kunden angewendet. Temporäre Felder wie Vorname, Nachname, Adresse, Postleitzahl und Umsatzsteuer-Identifikationsnummer werden zunächst schrittweise erfasst und können bei Bedarf angepasst werden. Erst nachdem alle erforderlichen Felder vollständig ausgefüllt und eine automatisch generierte Zusammenfassung der Kundendaten explizit bestätigt wurde, erfolgt die dauerhafte Speicherung des Kunden in einer separaten Kundentabelle. Auch rechnungsbezogene Parameter wie Leistungsbeschreibung, Menge und Einzelpreis werden zunächst in „User\_Sessions“ zwischengespeichert. „User\_Sessions“ fungiert somit als transiente Datenschicht, die keine dauerhafte Persistenz besitzt, sondern ausschließlich der Dialogsteuerung und der Vorbereitung der eigentlichen Geschäftsobjekte dient.

Die dauerhaft gültigen Kundendaten werden in der Tabelle „Customer“ verwaltet, die als zentrale Kundendatenbank des Systems dient. Neue Kundendatensätze können sowohl über die Webanwendung als auch direkt über den Chatbot angelegt werden. In dieser Tabelle werden alle für die Rechnungsstellung relevanten Stammdaten gespeichert, darunter Vor- und Nachname, Firmenbezeichnung, vollständige Adresse, E-Mail-Adresse, Telefonnummer sowie die Umsatzsteuer-Identifikationsnummer. Nach Abschluss eines Neuanlage-Dialogs werden die zuvor in „User\_Sessions“ erfassten temporären Kundendaten in einen konsistenten Datensatz in „Customer“ überführt. Bestehende Kunden werden über ihre in der Tabelle hinterlegten Attribute identifiziert und können im Dialog ausgewählt werden, sodass die bereits gespeicherten Stammdaten ohne erneute Eingabe für die Rechnungserstellung genutzt werden.

Die eigentliche Rechnungslogik ist in den Tabellen „Rechnung“ und „Rechnungsposition“ abgebildet. Die Tabelle „Rechnung“ speichert alle kopfbezogenen Informationen einer Rechnung, insbesondere den zugehörigen Kunden, die Rechnungsnummer, das Rechnungsdatum, den Leistungszeitraum, die Fälligkeit der Forderung sowie die gewählte Zahlungsart. Die hierfür erforderlichen Daten stammen einerseits aus der Tabelle „Customer“, andererseits aus den während des Dialogs in „User\_Sessions“ gespeicherten Eingaben und ergänzenden Feldern, die direkt im Chat erfasst und an Airtable übergeben werden. Zusätzlich werden in „Rechnung“ aggregierte Beträge gespeichert, indem die Summe aller zugehörigen Rechnungspositionen aus der Tabelle „Rechnungsposition“ übernommen und als Grundlage für die Berechnung des Bruttobetrags einschließlich Umsatzsteuer verwen-

---

det wird. Diese Berechnungen erfolgen über Formel-Felder, die den Nettobetrag mit einem vordefinierten Steuersatz multiplizieren und auf zwei Nachkommastellen runden.

Die Tabelle „Rechnungsposition“ erfasst die einzelnen Positionen einer Rechnung. Für jede Position werden Daten wie die Leistungsbeschreibung, die Menge und der Einzelpreis gespeichert, die zuvor dialogbasiert erhoben und in „User\_Sessions“ zwischengespeichert wurden. Die Berechnung des Positionsbetrags erfolgt über ein Formel-Feld, indem die Menge mit dem numerischen Einzelpreis multipliziert wird. Da eine Rechnung typischerweise mehrere Positionen umfassen kann, werden für jede Position separate Datensätze angelegt. Die Zuordnung der einzelnen Rechnungspositionen zu einer Rechnung erfolgt über Referenzinformationen wie die Rechnungsnummer oder interne Identifikatoren innerhalb der Workflow-Logik, ohne dass zwingend explizite Verknüpfungsfelder im Airtable-Schema definiert werden müssen.

Der Datenfluss im Anwendungsfall „Rechnung neu erstellen“ beginnt mit dem Eingang einer WhatsApp-Nachricht, mit der der Nutzer den Erstellungsprozess startet. Zunächst wird eine neue Sitzung in „User\_Sessions“ angelegt oder eine bestehende Sitzung aktualisiert, indem die Telefonnummer gespeichert und die Zustandsvariablen initialisiert werden. Anschließend werden alle für die Rechnungserstellung erforderlichen Informationen dialogbasiert erhoben und in den dynamischen Feldern von „User\_Sessions“ zwischengespeichert, darunter Kundendaten, Leistungsbeschreibungen, Mengen, Einzelpreise, Rechnungsdatum, Leistungszeitraum und Zahlungsfrist. Sobald alle Kundendaten vollständig vorliegen und bestätigt wurden, überführt der Workflow die temporären Angaben in einen dauerhaften Datensatz in „Customer“, sofern ein neuer Kunde anzulegen ist. Bei bestehenden Kunden wird lediglich der entsprechende Datensatz referenziert.

Im nächsten Schritt erzeugt der Workflow einen neuen Eintrag in der Tabelle „Rechnung“ und legt parallel für jede erfasste Rechnungsposition einen Datensatz in „Rechnungsposition“ an, der die zuvor zwischengespeicherten Leistungsdaten übernimmt. Nach Abschluss der Positionserfassung werden in „Rechnung“ die relevanten Summen berechnet, indem die Beträge der zugeordneten Positionen aggregiert und um die Umsatzsteuer ergänzt werden. Auf Basis dieser Daten wird in Google Docs ein Rechnungsdokument auf Grundlage eines vordefinierten Templates generiert, das die aus Airtable stammenden Felder einfügt und anschließend als PDF exportiert wird. Das fertige Dokument wird dem Nutzer über den Chat zur Verfügung gestellt und kann für weitere Prozesse, etwa die Ablage oder den Versand, verwendet werden.

Der Datenfluss im Anwendungsfall „Dokument ablegen“ ist ebenfalls dialoggesteuert, fokussiert sich jedoch auf die strukturierte Archivierung von Dateien. Der Prozess wird durch das Hochladen eines Dokuments, beispielsweise eines Belegs oder einer externen Rechnung, über den WhatsApp-Chatbot initiiert. Im Dialog werden die zur Zuordnung notwendigen Informationen erfasst, insbesondere der betroffene Kunde sowie der zeitliche Kontext der Ablage (Jahr, Quartal, Monat), und temporär in einer Sitzung gehalten. Auf Basis dieser Parameter erstellt der Workflow die entsprechende Ordnerstruktur in Google Drive, sofern diese noch nicht existiert, und lädt das Dokument in den kundenbezogenen Unterordner hoch.

Der dabei erzeugte Dateilink steht dem System für nachgelagerte Schritte zur Verfügung, etwa zur Anzeige im Kundenportal. Durch diesen Ablauf wird sichergestellt, dass hochgeladene Dokumente konsistent, nachvollziehbar und eindeutig in Bezug auf Kunde und Zeitraum zugeordnet werden können, ohne dass manuell eine eigene Ablagestruktur gepflegt werden muss.

## 4.4 Prozessablauf vom Chatbot bis zur Ablage

### 4.4.1 Ablauf „Rechnung neu erstellen“

In diesem Unterabschnitt wird der Ablauf des Anwendungsfalls „Rechnung neu erstellen“ beschrieben. Der Prozess beginnt mit der Interaktion der Nutzerinnen und Nutzer im WhatsApp-Chat. Anschließend werden die erforderlichen Rechnungsdaten schrittweise erfasst und gespeichert. Auf dieser Grundlage wird ein Rechnungsdokument erstellt, das den Nutzern am Ende des Prozesses als PDF direkt im Chat zur Verfügung gestellt wird.

Der Ablauf beginnt, sobald der Nutzer nach der initialen Verifizierung im WhatsApp-Chat den Befehl „Start“ sendet und im angezeigten Hauptmenü die Option „Rechnung erstellen“ auswählt. Anschließend wird in der Tabelle `User_Sessions` ein Sitzungseintrag angelegt oder aktualisiert. Das Zustandsfeld `current_step` wird dabei auf den entsprechenden Wert gesetzt, um den weiteren Dialogverlauf zu steuern. Auf Basis dieses Sitzungszustands führt der Chatbot den Nutzer schrittweise durch den Prozess. Nach jeder Eingabe wird gezielt die nächste passende Frage gestellt und der aktuelle Zustand entsprechend aktualisiert.

---

Im ersten inhaltlichen Schritt entscheidet der Nutzer, ob ein neuer Kunde angelegt oder ein bestehender Kunde ausgewählt werden soll. Wird die Option zur Neuanlage gewählt, führt der Chatbot den Nutzer schrittweise durch die Erfassung der erforderlichen Stammdaten. Dabei wird jedes Datenfeld einzeln abgefragt und zunächst temporär in der Tabelle `User_Sessions` gespeichert. Am Beispiel des Vornamens wird dieser Wert nach der Eingabe in einem dynamischen Feld hinterlegt und anschließend durch eine Bestätigungsfrage („Ist der eingegebene Vorname korrekt?“) verifiziert. Abhängig von der Antwort wird entweder der nächste Dialogschritt eingeleitet oder die Eingabe erneut abgefragt. Dieses Vorgehen wird analog für weitere Stammdaten wie Nachname, Firmenname, Adresse, Postleitzahl, Ort, Land, E-Mail-Adresse, Telefonnummer sowie die Umsatzsteuer-Identifikationsnummer angewendet. Nachdem alle erforderlichen Angaben vollständig erfasst und bestätigt wurden, werden die gesammelten Daten aus `User_Sessions` in einen neuen, dauerhaften Datensatz in der Tabelle `Customer` überführt. Gleichzeitig werden die kopfbezogenen Kundendaten für die spätere Erstellung der Rechnungsvorlage vorbereitet.

Entscheidet sich der Nutzer für die Auswahl eines bestehenden Kunden, wird zunächst ein Suchbegriff abgefragt und in der Tabelle `User_Sessions` gespeichert. Auf Grundlage dieses Suchbegriffs werden passende Kundendatensätze ermittelt, woraufhin der Chatbot unterschiedliche Dialogpfade anbietet. Ergibt die Suche keinen Treffer, kann der Nutzer entweder einen neuen Kunden anlegen, einen weiteren Suchversuch starten oder den Vorgang abbrechen. Führt die Suche zu genau einem Ergebnis, kann dieser Kundendatensatz direkt übernommen oder eine erneute Suche ausgelöst werden. Werden mehrere passende Kunden gefunden, zeigt der Chatbot eine Auswahlliste an, aus der der gewünschte Datensatz ausgewählt werden kann. In allen Fällen führt eine erfolgreiche Kundenauswahl dazu, dass die entsprechende Kundenreferenz in `User_Sessions` hinterlegt wird. Zusätzlich werden die kopfbezogenen Kundendaten in der Tabelle `Rechnung` gesetzt, um den weiteren Rechnungsprozess vorzubereiten.

An die Kundenauswahl schließt sich die Erfassung der rechnungsbezogenen Daten an. Zunächst wird das Rechnungsdatum abgefragt. Dabei kann der Nutzer entweder das aktuelle Datum („heute“) auswählen oder ein eigenes Datum im Format `TT.MM.JJJJ` angeben. Die eingegebene Datumsangabe wird zunächst in `User_Sessions` gespeichert und anschließend in das entsprechende Datumsfeld der Tabelle `Rechnung` übernommen. Im nächsten Schritt wird die Rechnungsnummer im vorgegebenen Format erfasst und ebenfalls dort hinterlegt. Auf Grundlage dieser Angaben ist der Rechnungskopf vollständig beschrieben. Anschließend beginnt die Erfassung der einzelnen Rechnungspositionen.

Die Erfassung der Rechnungspositionen erfolgt dialogbasiert und unterstützt sowohl Texteingaben als auch Sprachnachrichten. Zunächst wird eine Leistungsbeschreibung abgefragt. Diese kann entweder direkt als Text eingegeben oder aus einer Sprachnachricht transkribiert werden. Optional kann eine KI-Schnittstelle genutzt werden, um aus einer frei formulierten Beschreibung eine prägnante und formal geeignete Rechnungsposition zu erzeugen. Das generierte Ergebnis wird dem Nutzer zur Bestätigung im Chat angezeigt. Nach der Bestätigung der Leistungsbeschreibung fragt der Chatbot nacheinander die Menge und den Einzelpreis ab. Die eingegebenen Werte werden jeweils temporär in `User_Sessions` gespeichert und durch kurze Rückfragen bestätigt. Sobald alle Angaben vollständig vorliegen, werden die Daten in einen neuen Datensatz in der Tabelle `Rechnungsposition` überführt. Anschließend kann der Nutzer entscheiden, ob weitere Rechnungspositionen erfasst werden sollen. Wird dies bestätigt, wiederholt sich der beschriebene Ablauf für die nächste Position.

Sind alle Rechnungspositionen erfasst, werden im letzten Schritt die Zahlungsbedingungen abgefragt. Dazu gehört zunächst die Auswahl der Zahlungsfrist, beispielsweise 7, 14 oder 30 Tage. Diese wird im entsprechenden Feld der Tabelle `Rechnung` gespeichert. Darüber hinaus wird die gewünschte Zahlart erfasst, etwa Barzahlung, Kartenzahlung oder Überweisung. Auch diese Information wird als weiteres Attribut in der Tabelle `Rechnung` hinterlegt. Auf Grundlage der erfassten Daten werden anschließend die relevanten Summen berechnet. Hierzu aggregiert Airtable die Beträge der verknüpften Datensätze aus der Tabelle `Rechnungsposition` und ergänzt diese um die Umsatzsteuer. Damit liegen alle für die Rechnung erforderlichen Informationen strukturiert in den Tabellen `Customer`, `Rechnung` und `Rechnungsposition` vor.

Auf Basis der in Airtable vorliegenden Daten wird abschließend ein Rechnungsdokument erstellt. Hierzu wird in Google Docs ein vordefiniertes Template geöffnet und mit den Werten aus den Tabellen `Rechnung`, `Rechnungsposition` und `Customer` befüllt. Dabei werden die im Dokument enthaltenen Platzhalter durch die entsprechenden Feldinhalte ersetzt. Das fertig befüllte Dokument wird anschließend als PDF exportiert. Zum Abschluss erhält der Nutzer im WhatsApp-Chat die fertige Rechnung in Form einer PDF-Datei. Damit ist der gesamte Prozess von der initialen Interaktion im Chat bis zur Bereitstellung des formalen Rechnungsdokuments vollständig automatisiert abgeschlossen.

---

#### 4.4.2 Ablauf „Dokument ablegen“

Der Anwendungsfall „Dokument ablegen“ beschreibt den dialogbasierten Ablauf, mit dem Nutzerinnen und Nutzer Belege oder andere geschäftsrelevante Dokumente über den WhatsApp-Chat hochladen können. Ziel dieses Prozesses ist die strukturierte Ablage der Dokumente, sodass sie später zuverlässig wiedergefunden werden können. Ähnlich wie beim Anwendungsfall zur Rechnungserstellung erfolgt der Einstieg über den Chatbot. Die weitere Verarbeitung der Daten sowie die Ablage der Dokumente werden durch die angebundenen Systeme im Hintergrund gesteuert.

Der Ablauf beginnt, sobald die Nutzerin oder der Nutzer den Chat mit dem Rechnungsbots öffnet und nach der initialen Verifizierung den Befehl „Start“ sendet. Daraufhin zeigt der Chatbot ein Hauptmenü mit den Optionen „Rechnung erstellen“ und „Dokument speichern“ an. Wählt der Nutzer die Option „Dokument speichern“, wird in der Tabelle `User_Sessions` ein entsprechender Sitzungseintrag angelegt oder aktualisiert. Gleichzeitig wird das Zustandsfeld `current_step` so gesetzt, dass der folgende Dialog eindeutig dem Anwendungsfall „Dokument ablegen“ zugeordnet ist.

Im ersten Schritt der Dokumentenablage wird der zeitliche Kontext für die spätere Ablagestruktur festgelegt. Dazu fordert der Chatbot die Nutzerinnen und Nutzer auf, ein Jahr auszuwählen. In der Regel stehen dabei das aktuelle Jahr sowie die beiden unmittelbar vorangegangenen Jahre zur Auswahl, beispielsweise 2023, 2024 und 2025. Die getroffene Auswahl wird in der Tabelle `User_Sessions` in einem entsprechenden Feld, etwa `Jahr`, gespeichert. Gibt die Nutzerin oder der Nutzer eine ungültige Eingabe ein, die eine erneute Auswahl des Jahres angefordert. Dieses Vorgehen stellt sicher, dass für die spätere Ordnerbildung ausschließlich zulässige Werte verwendet werden.

Nachdem das Jahr erfolgreich festgelegt wurde, fragt der Chatbot im nächsten Schritt das zugehörige Quartal ab. Hierzu werden die vier Quartale des Jahres in einer Auswahlliste angezeigt. Die getroffene Auswahl wird in der Tabelle `User_Sessions`, beispielsweise im Feld `Quartal`, gespeichert. Anschließend wird der Monat abgefragt, der innerhalb des zuvor gewählten Quartals liegt. Dabei ist die Auswahl auf die Monate des entsprechenden Quartals beschränkt. Auf diese Weise können die Nutzerinnen und Nutzer nur konsistente Kombinationen aus Jahr, Quartal und Monat angeben. Eingaben, die nicht der vorgegebenen Auswahl entsprechen, werden zurückgewiesen und führen zu einer erneuten Abfrage.

Sobald Jahr, Quartal und Monat ausgewählt wurden, liegen alle erforderlichen Metadaten für die spätere Ablagestruktur vor. Diese Informationen werden in der Tabelle `User_Sessions` gespeichert. Gleichzeitig wird der Zustandsparameter `current_step` auf den nächsten Verarbeitungsschritt gesetzt, der den Uploadvorgang einleitet. Anschließend fordert der Chatbot die Nutzerinnen und Nutzer dazu auf, das zu archivierende Dokument direkt im WhatsApp-Chat hochzuladen. Dabei kann es sich beispielsweise um eine externe Rechnung, einen Zahlungsbeleg oder ein anderes geschäftsrelevantes Dokument handeln.

Geht ein Dokument im Chat ein, wird die Datei vom Workflow entgegengenommen und der erfolgreiche Upload gegenüber der Nutzerin oder dem Nutzer bestätigt. Anschließend prüft das System, ob die erforderliche Ordnerstruktur in Google Drive bereits vorhanden ist. Die Ablage erfolgt in einer hierarchisch aufgebauten Struktur, die sich beispielsweise nach Kunde, Jahr, Quartal und Monat gliedert. Existiert ein Ordner für das ausgewählte Jahr bereits, wird dieser wiederverwendet. Andernfalls wird er neu angelegt. Gleiches gilt für die untergeordneten Ordner auf Quartals- und Monatsebene. Das hochgeladene Dokument wird abschließend im passenden Monatsordner des zuvor gewählten Jahres und Quartals gespeichert.

Durch diesen Ablauf wird das Dokument konsistent und nachvollziehbar in der vorgesehenen Ablagestruktur archiviert. Die in `User_Sessions` erfassten Parameter bilden gemeinsam mit der dynamischen Ordnererstellung in Google Drive die Grundlage für eine eindeutige zeitliche Zuordnung der Dokumente. Dadurch entfällt für die Nutzerinnen und Nutzer die manuelle Pflege von Verzeichnissen oder das Wissen über konkrete Dateipfade. Dies erleichtert sowohl das spätere Wiederfinden der Dokumente als auch die Nutzung der Ablage für nachgelagerte Prozesse. Dazu zählen beispielsweise die Zusammenarbeit mit Steuerberatungen oder interne Auswertungen.

## **4.5 Sicherheits- und Datenschutzkonzept**



## 5 Implementierung

### 5.1 Implementierung des Chatbots (make.com-Szenarien)

#### 5.1.1 Implementierung des Chatbots (Make.com-Szenarien)

Das in Kapitel 4 konzipierte Chatbot-System wurde auf der Automatisierungsplattform **Make.com** umgesetzt. Die Dialoglogik besteht aus mehreren Szenarien, die über Webhooks Ereignisse auslösen, Nachrichten verarbeiten und Daten zwischen WhatsApp, Airtable, Google Docs und Google Drive austauschen. Die Grundlage der Kommunikation bildet die **WhatsApp Business Cloud API**, deren eingehende Nachrichten ein *Watch Events*-Modul in Make als Trigger empfängt [10]. Jedes eingehende Ereignis löst genau einen Ablauf aus, sodass Nachrichten synchron verarbeitet werden.

#### Szenario-Start und Nutzerregistrierung

Nach Aktivierung durch ein eingehendes WhatsApp-Ereignis ruft das Szenario die zugehörige Telefonnummer ab und sucht in Airtable nach bestehendem Nutzerstatus. Über das Modul *Search records* der Airtable-App werden Datensätze in der Tabelle *User\_Sessions* abgefragt [9]. Wird kein Eintrag gefunden, legt das Szenario automatisch eine neue Sitzung an, speichert die Telefonnummer im Feld *phone\_number* und setzt *current\_step* auf *menu*. Anschließend erhält der Nutzer eine automatisierte Willkommensnachricht mit dem Hinweis, die Eingabe "Start" zu senden, um den Prozess zu beginnen. Dieses Messaging erfolgt über das Modul *Send a message* der WhatsApp-Integration [10].

## Dialogsteuerung und Routing

Die gesamte Interaktionslogik wird durch einen zentralen Router gesteuert, der anhand des Felds `current_step` in `User_Sessions` entscheidet, welcher Teilablauf aktiviert wird. Sobald ein Nutzer beispielsweise “Start” sendet, reagiert der Pfad *menu*: Es wird eine Nachricht mit zwei interaktiven Buttons gesendet – “Rechnung erstellen” und “Dokument speichern”. Drückt der Nutzer einen dieser Buttons, aktualisiert das Szenario das Statusfeld `current_step` in `Airtable` und verzweigt in den passenden Ablauf. Zur Verarbeitung mehrerer möglicher Folgeschritte kommen in Make Router-Module und Filterbedingungen zum Einsatz, die unterschiedliche Ausgänge abhängig von Zustands- und Eingabewerten aktivieren [3].

### Ablauf “Rechnung erstellen”

Im Pfad “**Rechnung erstellen**” beginnt die Datenerfassung mit der Kundenauswahl. Der Benutzer kann wählen, ob ein neuer Kunde angelegt oder ein bestehender gesucht werden soll. Bei Auswahl *Kunde suchen* fordert der Chatbot einen Suchbegriff an, speichert diesen in `User_Sessions` und durchsucht die `Airtable`-Tabelle `Customer` nach passenden Einträgen. Die Suchabfrage nutzt eine OR-Kombination mehrerer Felder (Name, Firmenname, Adresse u. a.) sowie die Funktion `LOWER()` zur case-insensitiven Suche [9]. Ein *Array aggregator* fasst die Ergebnisse zu einem Array zusammen, dessen Länge über nachgelagerte Filtermodule bestimmt, ob kein, ein oder mehrere Treffer vorliegen [3].

- **Kein Treffer:** Der Nutzer kann abbrechen, neu suchen oder einen neuen Kunden erstellen.
- **Ein Treffer:** Der Kunde kann bestätigt oder eine erneute Suche ausgelöst werden.
- **Mehrere Treffer:** Der Nutzer wählt per nummeriertem Emoji den gewünschten Datensatz aus.

Nach Auswahl oder Neuanlage eines Kunden werden die zugehörigen Stammdaten in `Customer` gespeichert und zugleich als Relation in Rechnung übernommen. Der folgende Dialog sammelt schrittweise rechnungsbezogene Informationen (Datum, Rechnungsnummer, Leistungsbeschreibung, Menge, Einzelpreis, Zahlungsfrist, Zahlart). Die Dialogschritte sind durch `current_step`-Werte (z. B. *Datum ausgewählt*, *Position erstellt*)

---

klar strukturiert. Eingaben werden jeweils kurz bestätigt, bevor das Szenario in den nächsten Schritt wechselt.

Für Freitexte wie Leistungsbeschreibungen können sowohl Texteingaben als auch Sprachnachrichten genutzt werden. Audiodaten werden mit einem *Download media*-Modul der WhatsApp-Integration heruntergeladen und anschließend durch einen KI-Dienst (z. B. ChatGPT) transkribiert. Auf Basis des transkribierten Textes erzeugt ein weiteres KI-Modul eine kurze, formal geeignete Rechnungsposition. Ein restriktiv formuliertes Prompt begrenzt dabei Satzlänge, Wortanzahl und untersagt das Hinzufügen nicht genannter Inhalte, um eine präzise und sachliche Beschreibung sicherzustellen. Das Ergebnis wird dem Nutzer im Chat zur Bestätigung angezeigt; bei Ablehnung kann eine neue Eingabe oder Aufnahme gestartet werden.

Nach der Positionsbestätigung fragt der Bot nach Menge und Einzelpreis. Beide Werte werden verifiziert, in temporären Feldern von *User\_Sessions* zwischengespeichert und anschließend in den Tabellen *Rechnung* und *Rechnungsposition* persistiert. Abschließend erstellt das Szenario mit der Google-Docs-Integration ein Rechnungsdokument auf Basis eines Templates, fügt alle dynamischen Felder ein, exportiert das Dokument als PDF und sendet es dem Nutzer direkt über WhatsApp zurück [8]. Damit ist der End-to-End-Prozess der Rechnungserstellung abgeschlossen.

### Ablauf “Dokument speichern”

Wenn im Hauptmenü stattdessen *Dokument speichern* gewählt wird, läuft der zweite zentrale Pfad: die strukturierte **Dokumentenablage**. Der Chatbot fragt Jahr, Quartal und Monat nacheinander ab, speichert die Antworten in *User\_Sessions* und fordert anschließend zum Hochladen des Dokuments auf. Auf diese Weise wird der zeitliche Kontext der Ablage eindeutig bestimmt und nutzerseitig validiert.

Das Szenario prüft anschließend mit einem *Search files/folders*-Modul, ob im Google Drive bereits passende Ordner bestehen. Falls nicht, werden diese in der Reihenfolge Telefonnummer → Jahr → Quartal → Monat automatisch erstellt. Nach erfolgreicher Prüfung oder Anlage lädt das Modul *Upload a file* die Datei in den entsprechenden Ordner und schreibt den resultierenden Drive-Link zurück nach *Airtable*, sodass das Dokument später über die Web-Anwendung konsistent gefunden und abgerufen werden kann [11].

## 5.2 Datenhaltung und Strukturen in Airtable

## 5.3 Automatische Dokumentenerstellung in Google Docs

## 5.4 Ablagestrategie in Google Drive (Jahr/Quartal/Monat)

## 5.5 Implementierung der ClientHub WebApp

### 5.5.1 Frontend (React, UI-Konzept)

### 5.5.2 Edge Functions in Lovable Cloud

### 5.5.3 API-Anbindung an Airtable und Google Drive

## 6 Evaluation

### 6.1 Testkonzept und Testumgebung

### 6.2 Funktionale Tests

### 6.3 Performance-Analyse

### 6.4 Vergleich: manueller vs. automatisierter Prozess

### 6.5 Usability-Bewertung

### 6.6 Grenzen des Systems und Risiken

## 7 Fazit und Ausblick

### 7.1 Zusammenfassung der Ergebnisse

### 7.2 Beantwortung der Forschungsfragen

### 7.3 Beitrag der Arbeit und praktische Implikationen

### 7.4 Ausblick und Weiterentwicklung

# Literaturverzeichnis

- [1] BALZERT, Helmut: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum, 2011
- [2] BECK-IT: *Die manuelle und automatisierte Rechnungsverarbeitung im Vergleich*. 2022. – URL <https://beck-it.com/news/die-manuelle-und-automatisierte-rechnungsverarbeitung-im-vergleich/>. – Zugriffsdatum: 2025-12-13
- [3] CONSULTIVO: *Mastering Make.com Array Aggregator*. 2025. – URL <https://consultivo.com/make-com-array-aggregator-guide/>. – Zugriffsdatum: 2025-12-26
- [4] DPUNKT.VERLAG: *Grundlagen des modularen Softwareentwurfs*. 2020. – URL [https://www.assets.dpunkt.de/openbooks/Openbook\\_Spichale\\_API-Design\\_2A.pdf](https://www.assets.dpunkt.de/openbooks/Openbook_Spichale_API-Design_2A.pdf). – Zugriffsdatum: 2025-12-13
- [5] DPUNKT.VERLAG: *Langlebige Softwarearchitekturen – Erweiterungen*. 2020. – URL <https://www.langlebige-softwarearchitekturen.de/media/wps/uploads/langlebige-softwarearchitekturen-erweiterungen.pdf>. – Zugriffsdatum: 2025-12-13
- [6] FLOWWER: *Automatisierte Rechnungsverarbeitung: Effizient & fehlerfrei*. 2025. – URL <https://www.flowwer.de/automatisierte-rechnungsverarbeitung-der-beste-weg-zum-erfolg/>. – Zugriffsdatum: 2025-12-13
- [7] JOHANN, Peter: *Nicht-funktionale Anforderungen*. 2025. – URL <https://www.peterjohann-consulting.de/nicht-funktionale-anforderungen/>. – Zugriffsdatum: 2025-12-13
- [8] MAKE.COM: *Google Docs Integration — Workflow Automation*. 2024. – URL <https://www.make.com/en/integrations/auto-content-api/google-docs>. – Zugriffsdatum: 2025-12-26

- [9] MAKE.COM: *Airtable Modules — Search records*. 2025. – URL <https://apps.make.com/airtable-modules>. – Zugriffsdatum: 2025-12-24. – Abruf am 24.12.2025
- [10] MAKE.COM: *WhatsApp Business Cloud — Modules Documentation*. 2025. – URL <https://apps.make.com/whatsapp-business-cloud-modules>. – Zugriffsdatum: 2025-12-24. – Abruf am 24.12.2025
- [11] MAKE.COM COMMUNITY: *How To Connect Make.com to Google Drive (2025 Tutorial)*. 2025. – URL <https://www.youtube.com/watch?v=AiUT9cOCmVU>. – Zugriffsdatum: 2025-12-26. – YouTube-Tutorial zu Google-Drive-Anbindung
- [12] MEDIA, Seibert: *Qualität, funktionale und nichtfunktionale Anforderungen in der Softwareentwicklung*. 2018. – URL <https://seibert.group/blog/2018/05/14/qualitaet-funktionale-und-nichtfunktionale-anforderungen-in-in-der-software-entwicklung/>. – Zugriffsdatum: 2025-12-13
- [13] PLEO: *Automatisierte Rechnungsverarbeitung: Warum sie durch die E-Rechnung immer wichtiger wird*. 2024. – URL <https://blog.pleo.io/de/automatisierte-rechnungsverarbeitung>. – Zugriffsdatum: 2025-12-13
- [14] SPICHALE, Kai: *API-Design: Grundlagen und Best Practices für moderne Schnittstellen*. 2020. – URL [https://www.assets.dpunkt.de/openbooks/Openbook\\_Spichale\\_API-Design\\_2A.pdf](https://www.assets.dpunkt.de/openbooks/Openbook_Spichale_API-Design_2A.pdf). – Zugriffsdatum: 2025-12-13



# A Anhang

## A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
L <sup>A</sup> T <sub>E</sub> X	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments

## **Erklärung zur selbständigen Bearbeitung**

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe. Die in meiner Arbeit verwendeten KI-basierten Hilfsmittel habe ich (ggf. mit Produktnamen) angegeben.

Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschungsversuch behandelt werden können.

---

Ort

Datum

Unterschrift im Original