

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

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 Wirtschaftsinformatik*  
der Fakultät Informatik und digitale Gesellschaft  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer 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>viii</b>
<b>Tabellenverzeichnis</b>	<b>ix</b>
<b>Abkürzungen</b>	<b>x</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Motivation und Zielsetzung . . . . .	2
1.3 Forschungsfragen . . . . .	3
1.4 Vorgehensweise und Aufbau der Arbeit . . . . .	4
<b>2 Theoretische Grundlagen</b>	<b>6</b>
2.1 Digitalisierung administrativer Prozesse und Rechnungswesen . . . . .	6
2.2 Chatbots und dialogbasierte Systeme . . . . .	9
2.3 Cloudbasierte Datenhaltung und Integrationen . . . . .	11
2.4 Dokumentengenerierung in der Cloud . . . . .	12
2.5 Grundlagen moderner Webanwendungen . . . . .	14
<b>3 Technologischer Hintergrund</b>	<b>16</b>
3.1 Make.com als Integrations- und Automatisierungsplattform . . . . .	16
3.2 Airtable als Datenhaltung . . . . .	19
3.3 Google Docs und Google Drive . . . . .	20
3.4 Lovable Cloud / Supabase Edge Functions . . . . .	21
3.4.1 Eingesetzte Edge Functions und Zweck . . . . .	22
3.4.2 Aufgerufene Application Programming Interfaces (APIs) . . . . .	22
3.4.3 Zentrale Logik in den Edge Functions . . . . .	22
3.4.4 Begründung für Edge Functions . . . . .	23
3.4.5 Deployment und Konfiguration . . . . .	24
3.4.6 Sicherheitsaspekte . . . . .	24

3.4.7	Einordnung: Performance im Vergleich zu klassischem Backend . . . . .	24
3.5	Frontend-Stack (React, TypeScript, Tailwind, shadcn/ui) . . . . .	25
3.5.1	Grundlegende Technologien . . . . .	25
3.5.2	Routing und Zustandsverwaltung . . . . .	25
3.5.3	Formularverarbeitung und Validierung . . . . .	26
3.5.4	UI-Komponenten und Styling . . . . .	27
3.5.5	Struktur und Datenfluss im Frontend . . . . .	27
3.5.6	Responsives Design und Einschränkungen . . . . .	27
3.5.7	Deployment des Frontends . . . . .	28
<b>4</b>	<b>Konzeption des Systems</b>	<b>29</b>
4.1	Anforderungen und Use Cases . . . . .	29
4.1.1	Anwendungsfall „Rechnung neu erstellen“ . . . . .	31
4.1.2	Anwendungsfall „Dokument ablegen“ . . . . .	32
4.2	Zielarchitektur und Systemübersicht . . . . .	33
4.3	Datenmodell und Datenflüsse . . . . .	36
4.4	Prozessablauf vom Chatbot bis zur Ablage . . . . .	40
4.4.1	Ablauf „Rechnung neu erstellen“ . . . . .	40
4.4.2	Ablauf „Dokument ablegen“ . . . . .	44
4.5	Sicherheits- und Datenschutzkonzept . . . . .	46
<b>5</b>	<b>Implementierung</b>	<b>49</b>
5.1	Implementierung des Chatbots (make.com-Szenarien) . . . . .	49
5.2	Datenhaltung und Web-Anwendung (Airtable und ClientHub) . . . . .	53
5.2.1	Tabellenstruktur und Datenmodell in Airtable . . . . .	54
5.2.2	Interaktion zwischen Chatbot und Datenhaltung . . . . .	55
5.2.3	Architektur der Web-Anwendung (ClientHub) . . . . .	55
5.2.4	Funktionale Abgrenzung zwischen Web-App und Chatbot . . . . .	56
5.3	Automatische Dokumentenerstellung in Google Docs . . . . .	57
5.3.1	Rechnungsvorlage und Platzhalter . . . . .	58
5.3.2	Generierung und Versand der Portable Document Format (PDF)-Rechnung . . . . .	58
5.4	Ablagestrategie in Google Drive (Jahr/Quartal/Monat) . . . . .	59
5.5	Implementierung der ClientHub WebApp . . . . .	60
5.5.1	Frontend (React, UI-Konzept) . . . . .	60
5.5.2	Edge Functions in Lovable Cloud . . . . .	61

5.5.3 API-Anbindung an Airtable und Google Drive . . . . .	62
<b>6 Evaluation</b>	<b>63</b>
6.1 Testkonzept und Testumgebung . . . . .	63
6.2 Funktionale Tests . . . . .	64
6.3 Performance-Analyse . . . . .	66
6.4 Vergleich: manueller vs. automatisierter Prozess . . . . .	68
6.5 Usability-Bewertung . . . . .	70
6.6 Grenzen des Systems und Risiken . . . . .	71
<b>7 Fazit und Ausblick</b>	<b>73</b>
7.1 Zusammenfassung der Ergebnisse . . . . .	73
7.2 Beantwortung der Forschungsfragen . . . . .	74
7.3 Beitrag der Arbeit und praktische Implikationen . . . . .	76
7.4 Ausblick und Weiterentwicklung . . . . .	77
<b>Literaturverzeichnis</b>	<b>79</b>
<b>A Anhang</b>	<b>84</b>
A.1 Verwendete Hilfsmittel . . . . .	84
<b>Selbstständigkeitserklärung</b>	<b>86</b>

# Abbildungsverzeichnis

2.1	Generische Architektur eines dialogbasierten Chatbots. . . . .	10
2.2	Google Docs Vorlage mit Platzhaltern für automatisierte Rechnungserstellung. . . . .	13
3.1	Make.com-Szenario zur automatisierten Verarbeitung von WhatsApp-Nachrichten mit nachgelagerter Dokumentenerstellung und Ablage. . . . .	17
3.2	Airtable-Base mit tabellarischer Struktur zur Verwaltung von Dialogzuständen und Rechnungsdaten (Beispielansicht). . . . .	19
4.1	Use Cases des Systems: Rechnungserstellung und Dokumentenablage. . . . .	33
4.2	Zielarchitektur: WhatsApp Chatbot → Make.com → Airtable → Google Docs/Drive → React ClientHub. . . . .	34
4.3	Airtable-Datenmodell mit Relationen (User_Sessions, Customers, Rechnungen, Rechnungspositionen). . . . .	37
4.4	Prozessablauf des Anwendungsfalls „Rechnung neu erstellen“ vom dialogbasierten Start bis zur Generierung der PDF-Rechnung. . . . .	41
4.5	Prozessablauf des Anwendungsfalls „Dokument ablegen“ vom Chatbot bis zur strukturierten Ablage in Google Drive. . . . .	44
5.1	Hauptszenario des Chatbots in Make.com: WhatsApp-Trigger → Router → Airtable/Google-Integration → Antwort. . . . .	50
5.2	Router-Modul mit current_step-Steuerung: Verzweigung nach Dialogzustand. .	51
5.3	User_Sessions-Tabelle mit current_step-Steuerung und Telefonnummern. .	54
5.4	ClientHub Web-App: Kundenübersicht mit Such-/Filterfunktion. . . . .	57
5.5	ClientHub Dashboard mit Kunden- & Drive-Übersicht. . . . .	61
6.1	Subjektiv wahrgenommene Durchlaufzeiten der zentralen Use-Cases. . . . .	67

# Tabellenverzeichnis

5.1	ClientHub Frontend . . . . .	61
5.2	Edge Functions . . . . .	62
A.1	Verwendete Hilfsmittel und Werkzeuge . . . . .	84

# Abkürzungen

**API** Application Programming Interface.

**CORS** Cross-Origin Resource Sharing.

**CRUD** Create, Read, Update, Delete.

**Deno** Deno Runtime.

**DSGVO** Datenschutz-Grundverordnung.

**ERP** Enterprise Resource Planning.

**HTTP** Hypertext Transfer Protocol.

**JWT** JSON Web Token.

**OAuth** Open Authorization.

**PDF** Portable Document Format.

**REST** Representational State Transfer.

**SPA** Single Page Application.

**TS** TypeScript.



# 1 Einleitung

Die manuelle Rechnungs- und Dokumentenverarbeitung ist insbesondere für kleine Dienstleistungsunternehmen mit erhöhtem Zeitaufwand und einer erhöhten Fehleranfälligkeit verbunden [11]. Diese Arbeit entwickelt einen Chatbot-basierten Ansatz zur Digitalisierung, der Medienbrüche reduziert und administrative Prozesse automatisiert. Zunächst wird die Problemstellung skizziert, bevor Motivation, Ziele und Forschungsfragen erläutert werden.

## 1.1 Problemstellung

Kleine Dienstleistungsunternehmen und selbstständige Erwerbstätige verfügen häufig über keine spezialisierte Buchhaltungs- oder Enterprise Resource Planning (ERP)-Software, sondern erstellen ihre Rechnungen mit einfachen Office-Werkzeugen wie Textverarbeitungs- oder Tabellenkalkulationsprogrammen [8]. Kundendaten, Leistungsbeschreibungen sowie Beträge werden dabei manuell in Vorlagen übertragen. Die Rechnungen werden anschließend als PDF exportiert und per E-Mail oder Messenger an die Kundschaft versendet. Die Ablage der erzeugten Rechnungen und weiterer administrativer Dokumente erfolgt oftmals in unsystematischen Ordnerstrukturen auf lokalen Rechnern oder in allgemeinen Cloud-Speichern.

Diese Vorgehensweise führt zu mehreren Problemen. Zum einen ist die wiederkehrende manuelle Erstellung und Ablage von Rechnungen zeitaufwendig und fehleranfällig, etwa durch Tippfehler, fehlende oder unvollständige Pflichtangaben sowie uneinheitliche Formatierungen. Zum anderen entstehen Medienbrüche zwischen Kommunikationskanälen, Office-Dokumenten und Ablagesystemen, da Informationen aus E-Mails, Chats oder Notizen in unterschiedlichen Systemen erneut eingegeben und gepflegt werden müssen. Dies erschwert die spätere Nachvollziehbarkeit von Belegen, die strukturierte Vorbereitung von Unterlagen für Steuerberatung und Finanzverwaltung sowie den schnellen Zugriff auf vergangene Rechnungen und relevante Dokumente [11, 38].

Gleichzeitig stehen einfache cloudbasierte Werkzeuge zur Verfügung, mit denen sich Datenhaltung, Dokumentenerzeugung und Kommunikation grundsätzlich integrieren ließen. Viele kleine Unternehmen nutzen solche Dienste jedoch isoliert und ohne durchgängige Automatisierung, etwa indem sie zwar Cloud-Speicher oder Online-Textverarbeitung einsetzen, die dazwischenliegenden Prozesse aber weiterhin manuell ausführen. Es fehlt eine niedrigschwellige Lösung, die an den alltäglichen Arbeitsgewohnheiten der Zielgruppe ansetzt, Medienbrüche reduziert und sowohl die Erstellung als auch die strukturierte Ablage von Rechnungen und administrativen Dokumenten unterstützt, ohne den Einsatz komplexer und kostenintensiver Buchhaltungssysteme zu erfordern [19].

## 1.2 Motivation und Zielsetzung

Die in Abschnitt 1.1 beschriebene Ausgangslage zeigt, dass viele kleine Dienstleistungsunternehmen und Selbstständige zwar grundlegende digitale Werkzeuge nutzen, ihre administrativen Prozesse jedoch weiterhin stark manuell und medienbruchbehaftet organisieren. Rechnungen werden häufig erst mit zeitlichem Abstand zur erbrachten Leistung erstellt, die notwendigen Informationen müssen aus E-Mails, Notizen oder Gesprächen zusammengesucht werden, und die Ablage der entstandenen Dokumente erfolgt ohne konsistente Struktur. Dies führt nicht nur zu vermeidbarem Zeitaufwand und Fehlern, sondern erschwert auch die transparente Vorbereitung von Unterlagen für Steuerberatung und Finanzverwaltung.

Vor diesem Hintergrund besteht die Motivation dieser Arbeit darin, einen niedrigschwälligen Ansatz zur Digitalisierung der Rechnungs- und Dokumentenverarbeitung zu untersuchen, der sich an den realen Arbeitsgewohnheiten der Zielgruppe orientiert. Statt eine weitere, eigenständige Fachanwendung einzuführen, soll ein System entworfen werden, das an einen bereits etablierten Kommunikationskanal anknüpft und administrative Tätigkeiten dialogbasiert unterstützt. Durch die Kombination eines Chatbots mit einer cloudbasierten Datenhaltung und einer ergänzenden Web-Anwendung sollen Medienbrüche reduziert, wiederkehrende Arbeitsschritte automatisiert und eine strukturierte, nachvollziehbare Ablage der entstehenden Dokumente ermöglicht werden.

Ziel der Arbeit ist es, ein prototypisches System zu konzipieren und zu implementieren, das die Erstellung von Rechnungen und die Ablage administrativer Dokumente für kleine Unternehmen und Selbstständige unterstützt. Konkret soll der Prototyp es ermöglichen, Rechnungsdaten dialoggeführt über einen Chatbot zu erfassen, auf Basis eines Templates

automatisiert Rechnungsdokumente zu erzeugen und diese in einer konsistenten Struktur in der Cloud abzulegen. Ergänzend soll eine Web-Anwendung Kunden, Rechnungen und Dokumente übersichtlich darstellen und verwalten.

Die Arbeit entstand im Rahmen der Unterstützung eines selbstständigen Schneiders aus Winterhude, der etwa dreimal im Monat Rechnungen erstellen muss. Für ihn als Nicht-Muttersprachler fällt die manuelle Erstellung mit korrekter Formatierung, Pflichtangaben und Ablage besonders schwierig, da die deutsche Fachsprache im Rechnungswesen und die ständige Suche nach Vorlagen und Kundendaten viel Zeit und Frustration verursachen. Das entwickelte System adressiert genau diese praktischen Herausforderungen einer typischen Zielgruppenperson.

Die Umsetzung soll zeigen, inwieweit sich mit verhältnismäßig einfachen, cloudbasierten Bausteinen ein durchgängiger, medienarmer Prozess realisieren lässt und welche Grenzen und Verbesserungspotenziale sich im praktischen Einsatz eines solchen Ansatzes ergeben.

### **1.3 Forschungsfragen**

Aus der in Abschnitt 1.1 beschriebenen Ausgangslage sowie der in Abschnitt 1.2 formulierten Zielsetzung ergeben sich die folgenden Forschungsfragen, die im Rahmen dieser Arbeit untersucht werden:

1. Wie lässt sich ein prototypisches System zur Erstellung und strukturierten Ablage von Rechnungen und administrativen Dokumenten für kleine Dienstleistungsunternehmen und Selbstständige auf Basis gängiger cloudbasierter Werkzeuge konzipieren?
2. Inwieweit kann ein Chatbot, der an einen bestehenden Kommunikationskanal angebunden ist, die dialogbasierte Erfassung von Rechnungs- und Dokumentendaten unterstützen, sodass Medienbrüche reduziert und wiederkehrende manuelle Arbeitsschritte verringert werden?
3. Wie kann eine ergänzende Web-Anwendung gestaltet werden, die einen strukturierten Zugriff auf Kunden, Rechnungen und Dokumente ermöglicht und die im Chat erfassten Daten für die weitere Verwaltung nutzbar macht?

4. Inwieweit zeigen sich beim praktischen Einsatz des prototypisch umgesetzten Systems Grenzen, Herausforderungen und Verbesserungspotenziale in Bezug auf Nutzbarkeit, Prozessdurchlaufzeiten und technische Robustheit?

## 1.4 Vorgehensweise und Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in mehrere aufeinander aufbauende Schritte, die von der Einordnung des Themas über die Konzeption bis hin zur prototypischen Umsetzung und Evaluation des Systems reichen. Ziel ist es, die in den vorherigen Abschnitten formulierten Forschungsfragen systematisch zu bearbeiten und die dabei getroffenen Entscheidungen nachvollziehbar zu begründen.

Zunächst werden in Kapitel 2 die fachlichen und technischen Grundlagen vorgestellt, die für das Verständnis der Arbeit erforderlich sind. Dazu gehören insbesondere zentrale Konzepte der Rechnungs- und Dokumentenverarbeitung in kleinen Unternehmen sowie relevante Aspekte cloudbasierter Dienste und Integrationsplattformen. Auf dieser Basis werden zudem die in der Arbeit eingesetzten Technologien eingeordnet und begrifflich abgegrenzt.

Im anschließenden Kapitel wird die Konzeption des zu entwickelnden Systems beschrieben. Dabei werden die fachlichen Anforderungen präzisiert, zentrale Use-Cases herausgearbeitet und eine Zielarchitektur entworfen, die den Einsatz eines Chatbots, einer cloudbasierten Datenhaltung und einer ergänzenden Web-Anwendung miteinander verbindet. Die Konzeption umfasst sowohl die Gestaltung der Dialoge im Chat als auch die Struktur der zugrunde liegenden Daten und Dokumente.

Darauf aufbauend folgt die Implementierung des Prototyps. In diesem Kapitel wird erläutert, wie die zuvor konzipierte Architektur mit konkreten Werkzeugen und Diensten umgesetzt wird. Beschrieben werden die technische Integration der einzelnen Komponenten, die wichtigsten Implementierungsentscheidungen sowie ausgewählte Aspekte der praktischen Realisierung.

Im anschließenden Evaluationskapitel wird das entwickelte System anhand ausgewählter Szenarien untersucht. Dabei werden insbesondere die funktionale Vollständigkeit der Kernprozesse, die wahrgenommene Performance aus Nutzersicht sowie die Stärken und Grenzen des Ansatzes im praktischen Einsatz betrachtet. Zudem werden identifizierte Herausforderungen und mögliche Erweiterungen diskutiert.

## *1 Einleitung*

---

Abschließend fasst das Schlusskapitel die wesentlichen Ergebnisse der Arbeit zusammen, beantwortet die eingangs formulierten Forschungsfragen und gibt einen Ausblick auf weiterführende Arbeiten sowie potenzielle Weiterentwicklungen des prototypischen Systems.

## 2 Theoretische Grundlagen

In diesem Kapitel werden die fachlichen und technischen Grundlagen vorgestellt, die für das Verständnis der Arbeit erforderlich sind. Zunächst erfolgt eine Einführung in die Digitalisierung administrativer Prozesse sowie in spezifische Anforderungen des Rechnungswesens kleiner Unternehmen. Anschließend werden Chatbots und dialogbasierte Systeme beleuchtet, gefolgt von cloudbasierter Datenhaltung und Integrationsplattformen. Darauf aufbauend werden die Dokumentengenerierung in der Cloud sowie die Grundlagen moderner Webanwendungen als zentrale Bausteine der Lösung vorgestellt.

### 2.1 Digitalisierung administrativer Prozesse und Rechnungswesen

Die vorliegende Arbeit richtet sich nicht an Großunternehmen mit etablierten ERP- und Buchhaltungssystemen. Im Fokus stehen vielmehr kleine Dienstleistungsunternehmen und Einzelunternehmer, die ihre administrativen Aufgaben überwiegend ohne eigene Buchhaltungsabteilung bewältigen. Zu dieser Zielgruppe zählen insbesondere Freelancer sowie Kleinstbetriebe wie Handwerker, Berater, Coaches oder Fotografen. Darüber hinaus werden auch kleine Agenturen und Dienstleistungsbetriebe mit wenigen Mitarbeitenden berücksichtigt.

In diesen Unternehmen stützt sich das Rechnungswesen häufig auf einfache Office-Werkzeuge sowie wenig strukturierte Ablagesysteme [8, 9]. Viele administrative Tätigkeiten werden manuell durchgeführt und binden einen erheblichen Teil der verfügbaren Arbeitszeit. Für diese Zielgruppe stellt die Digitalisierung administrativer Prozesse eine zentrale Chance dar, da sie es ermöglicht, den manuellen Aufwand im Tagesgeschäft zu reduzieren und gleichzeitig die Qualität sowie die Nachvollziehbarkeit finanzieller Informationen nachhaltig zu verbessern.

Im Alltag nutzen kleine Unternehmen häufig Word- oder Excel-Vorlagen zur Erstellung von Angeboten und Rechnungen. Bestehende Dokumente werden dabei kopiert und manuell angepasst. Kundendaten, Rechnungsnummern, Datumsangaben sowie Beträge und Steuerinformationen müssen wiederholt per Hand eingetragen werden. Dieses Vorgehen ist zeitaufwendig und fehleranfällig. Aufgrund fehlender Systemunterstützung können Tippfehler, unvollständige Angaben oder doppelt vergebene Rechnungsnummern entstehen.

Die Kommunikation mit Kunden und Steuerberatungen erfolgt zudem überwiegend per E-Mail, wodurch häufig mehrere Versionen derselben Rechnung im Umlauf sind und zusätzlicher Abstimmungsaufwand entsteht. Belege liegen darüber hinaus in unterschiedlichen Formaten vor, etwa als Papierdokumente, Scans, Smartphone-Fotos oder PDF-Anhänge. Die Ablage erfolgt häufig unsystematisch in lokalen Ordnerstrukturen, Cloud-Speichern oder direkt im E-Mail-Postfach. Ein einheitliches Ordnungsprinzip nach Kunden oder Zeiträumen fehlt dabei oftmals.

Diese Arbeitsweise ist von zahlreichen Medienbrüchen geprägt, da Informationen manuell zwischen verschiedenen Anwendungen und Dokumenten übertragen werden müssen. Jeder Wechsel zwischen Medien und Systemen erfordert zusätzliche manuelle Eingriffe und erhöht die Fehleranfälligkeit. Gleichzeitig nimmt die Transparenz über den aktuellen Status von Rechnungen und Belegen ab. Insbesondere vor periodischen Stichtagen wie Monats- oder Jahresabschlüssen führt diese Situation zu zusätzlichem organisatorischem Aufwand und erhöhtem Stress, da Unterlagen für Steuerberaterinnen und Steuerberater häufig nachträglich zusammengestellt werden müssen. Verzögerte oder fehlerhafte Rechnungen wirken sich zudem unmittelbar auf die Liquidität aus, da Zahlungseingänge verspätet erfolgen und offene Posten schwerer nachzuvollziehen sind [11].

Die Digitalisierung administrativer Prozesse im Rechnungswesen verfolgt mehrere zentrale Ziele. Ein wesentliches Ziel ist die Verkürzung von Durchlaufzeiten. Wiederkehrende Prozessschritte wie das Erstellen, Versenden und Ablegen von Rechnungen sollen hierzu standardisiert und weitgehend automatisiert ablaufen. Darüber hinaus soll die Nachvollziehbarkeit verbessert werden, indem Rechnungen und zugehörige Dokumente zentral abgelegt und eindeutig Kunden sowie Zeiträumen zugeordnet werden. Dadurch sind sie jederzeit auswertbar, und die Transparenz über den Status von Belegen und Zahlungsvorgängen wird erhöht.

Ein weiterer Schwerpunkt liegt auf der Reduktion redundanter Dateneingaben. Kundendaten, Leistungsbeschreibungen und Zahlungsinformationen sollen nur einmal strukturiert

erfasst und anschließend in unterschiedlichen Kontexten wiederverwendet werden. Dazu zählen unter anderem die Rechnungserstellung, Auswertungen sowie die Vorbereitung für die Steuerberatung. Cloudbasierte Lösungen ermöglichen zudem einen orts- und zeitunabhängigen Zugriff auf Rechnungen und Belege. Dadurch werden sowohl die Zusammenarbeit mit externen Dienstleistern als auch mobile Arbeitsformen erleichtert [11, 9].

In Bezug auf das Szenario sind vor allem jene Informationen entscheidend, die erforderlich sind, um eine korrekte und nachvollziehbare Rechnungserstellung zu gewährleisten. Hierzu gehören konsistente Kundendaten wie Adresse und Umsatzsteuer-Identifikationsnummer, eindeutige Rechnungsnummern und Rechnungsdaten sowie strukturierte Leistungsbeschreibungen mit Mengen- und Preisangaben. Darüber hinaus sind steuerliche Kenngrößen und klar definierte Zahlungsbedingungen von Bedeutung.

Das in dieser Arbeit entworfene System adressiert diese Anforderungen bewusst als Vorstufe zur Finanzbuchhaltung. Es ermöglicht die strukturierte Erfassung von Rechnungsdaten, die zentrale Verwaltung von Kundenstammdaten, die automatisierte Erstellung von Dokumenten sowie eine revisionssichere Ablage der erzeugten Belege. Es werden weder Hauptbücher geführt noch Buchungssätze erstellt. Aufgaben wie die Kontierung, die Umsatzsteuervoranmeldung oder der Jahresabschluss verbleiben weiterhin bei Steuerberatungen und spezialisierten Buchhaltungssystemen. Diese können auf den konsistenten Beleg- und Stammdaten aufbauen, die das System bereitstellt, und die weiterführende buchhalterische Verarbeitung übernehmen.

Der in dieser Arbeit verfolgte Chatbot-basierte Ansatz unterscheidet sich grundlegend von klassischen, formularorientierten Rechnungsprogrammen, wie sie typischerweise über webbasierte Benutzeroberflächen bereitgestellt werden. Herkömmliche Systeme setzen voraus, dass Nutzer aktiv eine Anwendung öffnen, sich durch Masken und Menüs navigieren und eigenständig entscheiden, welche Felder auszufüllen sind. Der hier vorgestellte Ansatz verfolgt hingegen eine schrittweise, dialogbasierte Datenerfassung über einen bereits etablierten Kommunikationskanal. Nutzer interagieren mit dem System ähnlich wie mit einem Kontakt in einem Messenger und werden im Gesprächsverlauf gezielt durch den Erfassungsprozess geführt. Die selbstständige Bedienung komplexer Formulare ist dadurch nicht erforderlich.

Dieser Ansatz senkt insbesondere für technisch weniger affine Anwender die Einstiegshürde. Zudem ermöglicht er eine situative Datenerfassung direkt im Arbeitskontext, etwa unmittelbar nach Abschluss einer Dienstleistung. Geführte Eingaben und integrierte Validierungen tragen zusätzlich dazu bei, die Fehleranfälligkeit zu reduzieren. Studien

und Praxisberichte zur Automatisierung von Finanzprozessen, insbesondere bei KMU-Rechnungsprozessen, zeigen, dass digitale Systeme bei der Verarbeitung von Rechnungs- und Ausgabendaten Effizienzgewinne erzielen und gleichzeitig die Fehlerquote verringern können [11].

## 2.2 Chatbots und dialogbasierte Systeme

Chatbots sind Softwaresysteme, die mit Nutzenden über natürliche Sprache oder strukturierte Eingaben interagieren und dabei eine Konversation simulieren. Sie werden typischerweise in Messaging-Plattformen, Web-Chats oder Sprachassistenten eingebettet und dienen als Schnittstelle zu dahinterliegenden Informations- und Prozesssystemen. Abhängig von ihrer technischen Ausgestaltung reicht das Spektrum von einfachen, regelbasierten Antwortsystemen bis hin zu KI-basierten Assistenten. Letztere nutzen statistische Sprachmodelle, um Kontext zu berücksichtigen und flexibel auf unterschiedliche Eingaben reagieren zu können [6].

Früher waren Chatbots überwiegend regelbasiert aufgebaut. Sie arbeiteten mit fest definierten Dialogbäumen, Schlüsselwörtern und statischen Antwortbausteinen. Der Dialogverlauf wurde dabei durch If-Then-Regeln oder Zustandsautomaten gesteuert und konnte nur eine begrenzte Anzahl möglicher Interaktionen abbilden. Moderne dialogbasierte Systeme nutzen hingegen Verfahren der automatischen Sprachverarbeitung. Diese ermöglichen es, Benutzereingaben zu analysieren, Intentionen zu erkennen und relevante Entitäten wie Beträge, Datumsangaben oder Kundennamen zu extrahieren. Auf dieser Grundlage können solche Systeme komplexere Aufgaben übernehmen, etwa das Ausfüllen von Formularen, das Zusammenfassen von Informationen oder das Anstoßen nachgelagerter Geschäftsprozesse [6].

Aus architektonischer Sicht fungieren Chatbots als Vermittler zwischen einer Konversationsoberfläche und einer oder mehreren Backend-Anwendungen. Eingaben der Nutzenden werden dabei in strukturierte Informationen überführt, die zur Ansteuerung externer Dienste wie Datenbanken, Fachanwendungen oder Workflow-Systeme verwendet werden.

Abbildung 2.1 zeigt die generische Architektur eines dialogbasierten Chatbots als Vermittler zwischen Konversationsoberfläche und Backend-Diensten.

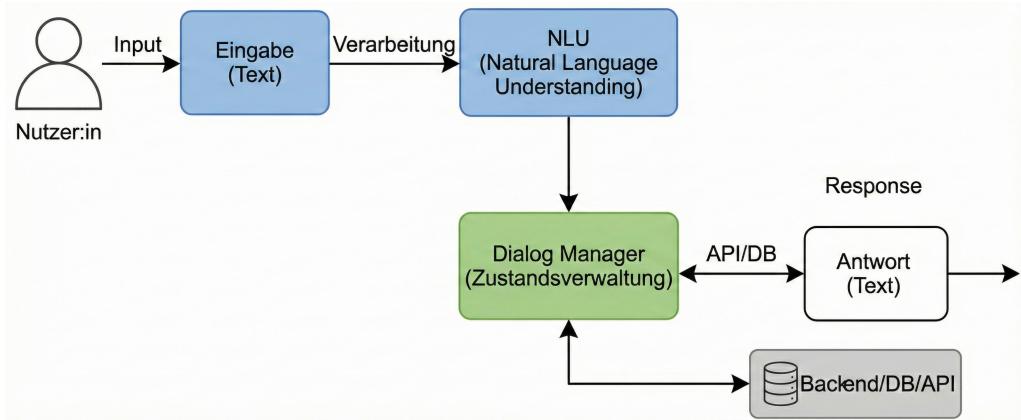


Abbildung 2.1: Generische Architektur eines dialogbasierten Chatbots.

Zur konsistenten Führung von Dialogen wird häufig ein expliziter Dialogzustand verwaltet. Dieser wird beispielsweise in einer Sitzungsdatenbank gespeichert und ermöglicht es, Kontexte über mehrere Nachrichten hinweg aufrechtzuerhalten. In geschäftskritischen Szenarien werden Chatbots zusätzlich mit Validierungsmechanismen, klar definierten Dialogflüssen und Eskalationspfaden kombiniert. Auf diese Weise lassen sich sowohl eine hohe Nutzerfreundlichkeit als auch die erforderliche Prozesssicherheit gewährleisten.

Im Finanz- und Rechnungswesen werden Chatbots zunehmend zur Unterstützung wiederkehrender und stark strukturierter Aufgaben eingesetzt. Dazu zählen unter anderem die Beantwortung typischer Rückfragen, die Erfassung standardisierter Informationen oder das Anstoßen definierter Geschäftsprozesse. Ein wesentlicher Vorteil dialogbasierter Systeme besteht darin, dass sie über etablierte Kommunikationskanäle wie Messenger-Anwendungen verfügbar sind. Dadurch lassen sie sich ohne umfangreiche Schulungsmaßnahmen in bestehende Arbeitsabläufe integrieren. Die Interaktion verlagert sich hierbei von komplexen Formularoberflächen hin zu einer geführten, schrittweisen Datenerfassung im Dialog [11].

Das in dieser Arbeit vorgestellte System nutzt diese Eigenschaften, indem ein Chatbot als zentrale, dialogbasierte Schnittstelle zur Erfassung von Rechnungs- und Dokumentendaten eingesetzt wird. Der Chatbot führt Nutzerinnen und Nutzer schrittweise durch den Erfassungsprozess, fragt fehlende Informationen gezielt ab und überprüft Eingaben unmittelbar auf Plausibilität. Die dabei gewonnenen strukturierten Daten werden anschließend an cloudbasierte Backend-Dienste übermittelt, dort gespeichert, für die automatisierte Dokumentenerzeugung verwendet und in nachgelagerte Prozessschritte

integriert. Auf diese Weise wird der Übergang von informeller Kommunikation, etwa einer kurzen Beschreibung der erbrachten Leistung per Text- oder Sprachnachricht, zu formalisierten Verwaltungsprozessen im Rechnungswesen weitgehend automatisiert und ohne Medienbrüche umgesetzt.

Daraus leitet sich für diese Arbeit die Anforderung ab, dass der Chatbot WhatsApp als etablierten Kommunikationskanal nutzt, einen regelbasierten Dialogzustand unter Verwendung einer Sitzungsdatenbank verwaltet und strukturierte Rechnungsdaten sicher extrahiert.

## 2.3 Cloudbasierte Datenhaltung und Integrationen

Cloudbasierte Datenhaltung bezeichnet die Speicherung von Informationen in extern betriebenen Rechenzentren, auf die über das Internet zugegriffen wird [30]. Für kleine Dienstleistungsunternehmen und Einzelunternehmer entfällt dadurch der Bedarf an eigener Serverinfrastruktur, während Daten orts- und geräteunabhängig verfügbar sind. Wartung, Ausfallsicherheit und Skalierung werden weitgehend von spezialisierten Anbietern übernommen, sodass sich die Unternehmen stärker auf ihr Kerngeschäft konzentrieren können.

Ein zentrales Merkmal cloudbasierter Systeme sind standardisierte Programmierschnittstellen. Über solche Web-APIs können unterschiedliche Anwendungen Daten lesen, schreiben und verarbeiten, ohne direkt auf die zugrunde liegende Infrastruktur zugreifen zu müssen. Web-Frontends, mobile Anwendungen und Automatisierungsdienste nutzen denselben Datenbestand, was Mehrfacherfassungen reduziert und konsistente Informationen in verschiedenen Nutzungskontexten ermöglicht.

Im Rechnungswesen kleiner Unternehmen erlaubt dieser Ansatz die Kombination spezialisierter Dienste statt eines monolithischen Systems. Eine cloudbasierte Datenbank kann etwa Kunden- und Rechnungsinformationen verwalten, während ein separater Speicherdienst für die Ablage von Dokumenten und eine Messaging-Plattform für die Kommunikation mit Nutzenden zuständig sind. Über Integrationslogik im Backend oder in Automatisierungsworflows werden diese Komponenten zu einem durchgängigen Gesamtsystem verknüpft, in dem Daten zwischen den Diensten ausgetauscht und in unterschiedlichen Oberflächen wiederverwendet werden.

Für das in dieser Arbeit betrachtete Szenario bedeutet dies, dass Stammdaten, Rechnungsinformationen und Verweise auf Dokumente zentral in der Cloud geführt werden, während ergänzende Dienste wie Dokumentengenerierung und Dateispeicherung eigenständig, aber angebunden betrieben werden. Chatbot und Webanwendung greifen auf dieselben Daten zu und stoßen über integrierte Schnittstellen fachliche Prozesse an. Dadurch entsteht eine flexible Architektur, in der einzelne Dienste bei Bedarf ersetzt oder erweitert werden können, ohne die Gesamtstruktur grundlegend ändern zu müssen.

## 2.4 Dokumentengenerierung in der Cloud

Unter Dokumentengenerierung in der Cloud wird die automatisierte Erstellung von Dateien wie Rechnungen, Angeboten oder Verträgen auf Basis digital vorliegender Daten verstanden. Anstatt Dokumente manuell in Textverarbeitungsprogrammen zu erstellen, werden strukturierte Informationen – etwa Kundenstammdaten, Positionslisten oder Zahlungsbedingungen – mit vordefinierten Vorlagen verknüpft. Die eigentliche Erzeugung des Dokuments erfolgt in einer Cloud-Umgebung, sodass keine lokale Bürossoftware erforderlich ist und die Erstellung jederzeit von verschiedenen Endgeräten ausgelöst werden kann.

Technisch basiert dieser Ansatz typischerweise auf Vorlagen, die Layout, Formatierung und feste Textbestandteile enthalten, während variable Inhalte über Platzhalter eingefügt werden.

Eine exemplarische Vorlage mit Platzhaltern ist in Abbildung 2.2 dargestellt.

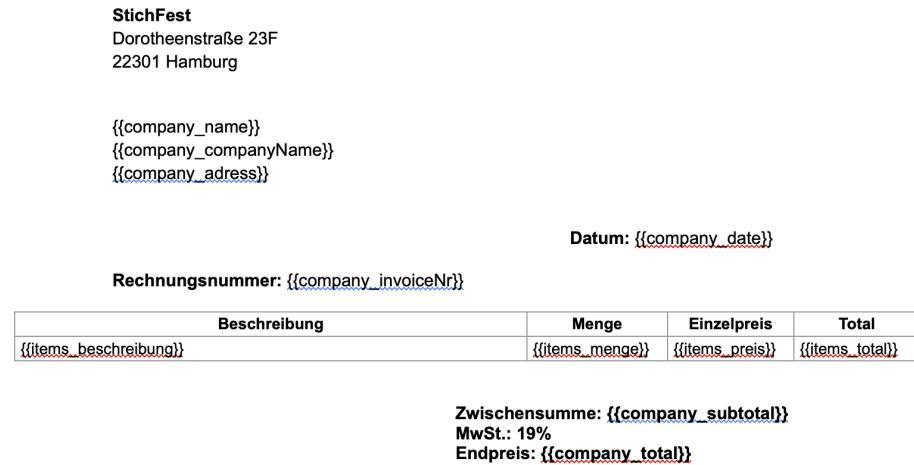


Abbildung 2.2: Google Docs Vorlage mit Platzhaltern für automatisierte Rechnungserstellung.

Bei der Generierung werden die Platzhalter durch konkrete Werte aus einem Datensystem ersetzt, wodurch aus einer einzigen Vorlage eine Vielzahl individueller Dokumente entstehen kann. Die Ausgabeformate reichen von bearbeitbaren Textdokumenten bis zu nicht veränderbaren Formaten wie PDF, die sich für den Versand an Kunden sowie für die revisionsorientierte Ablage eignen. Anpassungen am Erscheinungsbild oder an rechtlichen Pflichtangaben erfolgen zentral an der Vorlage und wirken sich unmittelbar auf alle zukünftigen Dokumente aus.

Die Cloud-Ausführung eröffnet zusätzliche Möglichkeiten der Integration in digitale Geschäftsprozesse. Dokumente können unmittelbar nach ihrer Erstellung automatisiert weiterverarbeitet werden, etwa durch Versand per E-Mail oder Messenger, Speicherung in strukturierten Ordnerhierarchien oder Übergabe an nachgelagerte Systeme wie Buchhaltung oder Dokumentenmanagement. Durch die enge Kopplung mit datenhaltenden Systemen und Workflows lassen sich manuelle Zwischenschritte deutlich reduzieren. Für kleine Unternehmen entsteht so ein durchgängiger Prozess von der Datenerfassung über die Dokumentenerzeugung bis zur Ablage, ohne dass Medienbrüche oder manuelle Formatisierungsarbeit erforderlich sind [10, 14].

Für das entwickelte System wird daher eine Google-Docs-Vorlage mit Platzhaltern verwendet, die automatisiert mit Daten aus Airtable befüllt und als PDF revisionssicher in Google Drive abgelegt wird.

## 2.5 Grundlagen moderner Webanwendungen

Moderne Webanwendungen unterscheiden sich deutlich von klassischen, statischen Websites. Während früher hauptsächlich einzelne HTML-Seiten ausgeliefert wurden, verarbeiten aktuelle Anwendungen komplexe Geschäftslogik, dynamische Daten und interaktive Benutzeroberflächen direkt im Browser. Grundlage ist dabei das Client–Server-Modell: Ein Webbrowswer fungiert als Client, der über das Hypertext Transfer Protocol (HTTP)-Protokoll Anfragen an einen Server sendet. Der Server verarbeitet diese Anfragen, greift bei Bedarf auf Datenquellen zu und liefert strukturierte Antworten zurück, die im Frontend dargestellt oder weiterverarbeitet werden.

Auf der Client-Seite kommen typischerweise JavaScript-Frameworks zum Einsatz, die Single-Page-Applications (Single Page Application (SPA)) ermöglichen. In einer SPA wird die Benutzeroberfläche nicht bei jedem Seitenwechsel vollständig neu geladen, sondern dynamisch im Browser aktualisiert. Daten werden über asynchrone Anfragen an serverseitige Schnittstellen geladen und in Komponenten gerendert. Dieses Architekturprinzip erlaubt reaktionsschnelle Oberflächen, die sich in ihrer Bedienung eher wie klassische Desktop-Anwendungen verhalten und komplexe Interaktionen, etwa Tabellen mit Filter- und Sortierungsfunktionen, komfortabel abbilden [32].

Die serverseitige Ebene moderner Webanwendungen stellt meist klar definierte Programmierschnittstellen bereit, häufig in Form von Representational State Transfer (REST)- oder JSON-basierten HTTP-APIs. Darüber werden fachliche Operationen wie das Lesen, Anlegen oder Aktualisieren von Datensätzen kapsuliert. Ergänzend gewinnen serverlose Architekturen und sogenannte Edge Functions an Bedeutung, bei denen einzelne Funktionsbausteine ereignisgesteuert und skalierbar in einer Cloud-Umgebung ausgeführt werden. Sicherheitsmechanismen wie Authentifizierung und Autorisierung, Transportverschlüsselung sowie rollenbasierte Zugriffskonzepte sind integraler Bestandteil solcher Anwendungen und sorgen dafür, dass sensible Daten nur von berechtigten Nutzenden eingesehen oder verändert werden können.

## *2 Theoretische Grundlagen*

---

Die ClientHub-Web-App nutzt dieses SPA-Prinzip mit React, TypeScript und Edge Functions für die strukturierte Darstellung von Kunden- und Rechnungsdaten aus Airtable sowie den Zugriff auf Drive-Dokumente.

# 3 Technologischer Hintergrund

Dieses Kapitel stellt die konkreten Technologien vor, die für die Umsetzung des Prototyps verwendet wurden. Zunächst wird Make.com als zentrale Integrations- und Automatisierungsplattform erläutert. Anschließend folgt Airtable als Datenhaltung, gefolgt von Google Docs und Google Drive zur Dokumentenerstellung und -ablage. Abschließend werden Lovable Cloud (Supabase Edge Functions) sowie der Frontend-Stack beschrieben.

## 3.1 Make.com als Integrations- und Automatisierungsplattform

Make.com ist eine cloudbasierte Integrations- und Automatisierungsplattform, die es ermöglicht, unterschiedliche Anwendungen, Dienste und Systeme miteinander zu verbinden. Die Umsetzung erfolgt ohne klassische Programmierung und basiert auf einem No-Code-beziehungsweise Low-Code-Ansatz. Workflows werden in einer grafischen Benutzeroberfläche modelliert, indem vordefinierte Funktionsbausteine konfiguriert und zu logischen Abläufen verknüpft werden. Dadurch können auch komplexe Integrationsprozesse ohne umfangreiche Programmierkenntnisse umgesetzt werden.

Aus technischer Sicht handelt es sich bei Make.com um eine Integrationsplattform, die externe Systeme über deren Web-APIs anbindet. Ereignisse und Datenflüsse werden dabei in einem zentralen Orchestrierungslayer zusammengeführt und automatisiert verarbeitet [25, 23].

Das zentrale Konstrukt in Make sind sogenannte Szenarien. Ein Szenario beschreibt einen vollständigen Automatisierungsablauf, der aus einem auslösenden Ereignis (Trigger), einer Abfolge von Verarbeitungsschritten sowie einer oder mehreren Aktionen besteht. Die einzelnen Bausteine eines Szenarios werden als Module bezeichnet. Diese repräsentieren entweder konkrete Anwendungsintegrationen, etwa zu WhatsApp, Airtable oder Google Docs, oder generische Funktionen wie Filter, Iteratoren, Aggregatoren oder Router.

### 3 Technologischer Hintergrund

---

Trigger-Module starten ein Szenario beispielsweise bei einer eingehenden Nachricht über einen Webhook. Aktionsmodule übernehmen anschließend die Verarbeitung der Daten, indem sie Informationen lesen, schreiben oder an andere Systeme übertragen.

Abbildung 3.1 zeigt ein exemplarisches Make.com-Szenario, das den zentralen Automatisierungsworkflow des Systems abbildet. Eingehende Nachrichten werden verarbeitet, strukturierte Daten in Airtable persistiert und darauf aufbauend Dokumente erzeugt und revisionssicher in Google Drive abgelegt.

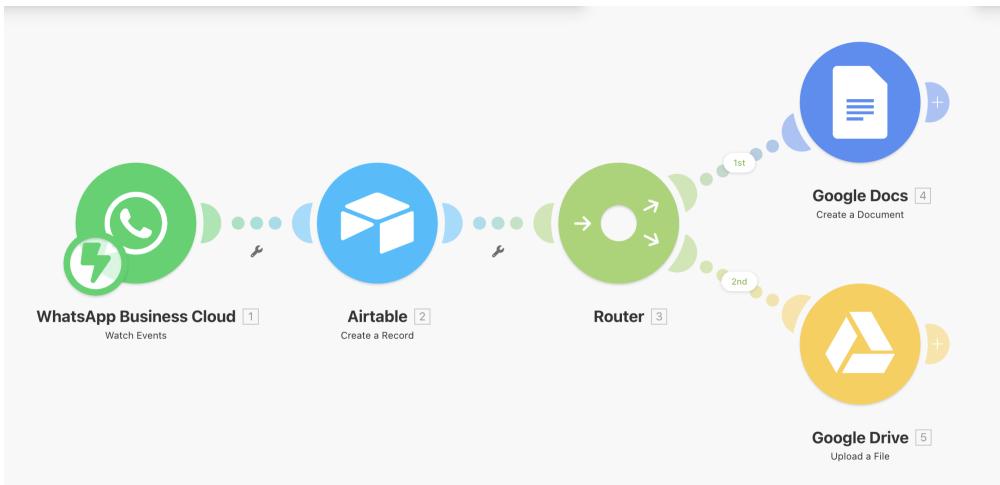


Abbildung 3.1: Make.com-Szenario zur automatisierten Verarbeitung von WhatsApp-Nachrichten mit nachgelagerter Dokumentenerstellung und Ablage.

Mithilfe visueller Router und Filter können alternative Verarbeitungswege modelliert werden, die abhängig von Dateninhalten oder Zuständen unterschiedliche Pfade innerhalb des Szenarios durchlaufen [23].

Die Plattform stellt darüber hinaus erweiterte Funktionen zur Ablaufsteuerung bereit. Mithilfe von Filtern können nur jene Datensätze weiterverarbeitet werden, die definierte Bedingungen erfüllen, etwa das Vorhandensein bestimmter Pflichtfelder. Iteratoren und Aggregatoren ermöglichen die Verarbeitung von Listenstrukturen. Dabei können Datensätze entweder einzeln durchlaufen oder zu zusammengefassten Ergebnissen aggregiert werden [23]. Router-Module erlauben es, Workflows in mehrere Pfade zu verzweigen, die jeweils eigene Bedingungen und nachgelagerte Verarbeitungsschritte besitzen. Auf diese Weise lassen sich auch komplexe und verzweigte Prozesse modellieren, ohne dass eigener Kontrollfluss-Code implementiert werden muss. Eine besondere Rolle nehmen

### *3 Technologischer Hintergrund*

---

Webhook-Trigger ein, da sie die nahezu Echtzeit-Verarbeitung von Ereignissen aus externen Systemen ermöglichen, sobald diese eine definierte URL aufrufen.

Im betrachteten Anwendungskontext fungiert Make.com als zentrale Vermittlungsschicht zwischen Chatbot, Datenhaltung sowie Dokumenten- und Speicherdiensten. Ereignisse aus der Kommunikationsschnittstelle werden über Trigger-Module empfangen, innerhalb von Szenarien verarbeitet und anschließend über Anwendungs-Module an Systeme wie Airtable, Google Docs oder Google Drive weitergeleitet. Die Plattform übernimmt die zentrale Steuerung der Datenflüsse und führt während der Verarbeitung notwendige Validierungen aus. Anschließend werden nachgelagerte Aktionen ausgelöst, etwa die automatische Erzeugung von Dokumenten oder die Aktualisierung von Datensätzen in angebundenen Systemen. Make.com fungiert damit als Orchestrierungsschicht zwischen Chatbot, Datenhaltung und Dokumentenservices. Insbesondere für kleinere Projekte und prototypische Anwendungen ergeben sich daraus mehrere Vorteile: Die initiale Implementierung kann ohne eigene Serverinfrastruktur erfolgen, vorgefertigte Konnektoren reduzieren den Integrationsaufwand, und die Prozesslogik lässt sich durch die visuelle Modellierung nachvollziehbar dokumentieren.

Gleichzeitig sind mit dem Einsatz einer Integrationsplattform wie Make.com auch Einschränkungen verbunden. Sehr umfangreiche Szenarien können an Übersichtlichkeit verlieren, da sämtliche Verarbeitungswege innerhalb eines visuellen Diagramms abgebildet werden müssen. Zudem ist die Abbildung komplexer Geschäftslogik nur eingeschränkt möglich. Darüber hinaus entsteht eine Abhängigkeit vom jeweiligen Anbieter, etwa hinsichtlich Verfügbarkeit, Preisgestaltung und Funktionsumfang der Plattform. Für hochskalierende oder besonders sicherheitskritische Anwendungen kann daher eine klassisch implementierte, individuell entwickelte Integrationsschicht sinnvoller sein.

Make.com stellt hingegen insbesondere für klein- bis mittelkomplexe Automatisierungsvorhaben mit begrenzten Ressourcen eine pragmatische und gut nachvollziehbare Lösung dar.

Zusammenfassend steuern Make.com-Szenarien den Workflow vom WhatsApp-Trigger über Airtable bis zur Dokumentenerzeugung in Google Docs sowie zur Ablage in Google Drive.

## 3.2 Airtable als Datenhaltung

Airtable ist eine cloudbasierte Plattform, die tabellenorientierte Benutzeroberflächen mit den Eigenschaften einer datenbankgestützten Anwendung kombiniert. Aus Anwendersicht ähnelt Airtable einer klassischen Tabellenkalkulation, erweitert diese jedoch um strukturierte Feldtypen, Relationen zwischen Tabellen sowie eine standardisierte Programmierschnittstelle. Dadurch eignet sich die Plattform insbesondere für Anwendungsszenarien, in denen Daten sowohl über eine webbasierte Oberfläche gepflegt als auch automatisiert von externen Systemen gelesen und geschrieben werden sollen [2].

Zentrale Konzepte in Airtable sind sogenannte Bases, die jeweils eine in sich geschlossene Datenbankinstanz mit mehreren Tabellen repräsentieren. Eine Tabelle besteht aus Datensätzen und Feldern, wobei die Felder unterschiedliche Typen annehmen können, etwa Text, Zahl, Datum, Auswahlfelder oder Verknüpfungen zu anderen Tabellen. Über verknüpfte Felder lassen sich einfache Relationen zwischen Tabellen modellieren, beispielsweise zwischen Kunden und zugehörigen Rechnungen.

Eine beispielhafte Tabellenstruktur der verwendeten Airtable-Base ist in Abbildung 3.2 dargestellt.

Nummern	User_Sessions	Customers	Rechnung	Rechngsposition	+	Tools
	Grid view				Felder ausblenden Filter Gruppierung Sortieren Farbe Teilen und synchronisieren	Teilen und synchronisieren
	A phone_number	A current_step	temp_customer_da...	nextStep	A suchbegriff	A Jahr
+						A Quartal

Abbildung 3.2: Airtable-Base mit tabellarischer Struktur zur Verwaltung von Dialogzuständen und Rechnungsdaten (Beispielansicht).

Darüber hinaus stehen verschiedene Sichten und Filter zur Verfügung, mit denen Teilmengen der Daten anhand frei definierbarer Kriterien angezeigt werden können. Formelfelder ermöglichen zudem die Berechnung dynamischer Werte [4]. Sie können unter anderem zur Aggregation von Beträgen oder zur Erzeugung formatierter Anzeigehinhalte eingesetzt werden.

Für den Einsatz als zentrale Datenhaltung in einem verteilten System ist insbesondere die von Airtable bereitgestellte REST-API von Bedeutung. Sie ermöglicht es, Datensätze programmatisch anzulegen, zu aktualisieren, zu löschen oder anhand definierter Filterkriterien abzufragen [3]. Auf diese Weise kann Airtable als zentrale „Single Source of Truth“

für fachliche Informationen fungieren. Unterschiedliche Frontends und Automatisierungsprozesse greifen dabei über die API auf denselben konsistenten Datenbestand zu. Durch die Kombination aus grafischer Benutzeroberfläche und Programmierschnittstelle eignet sich Airtable sowohl für die manuelle Pflege und Kontrolle von Daten als auch für die Integration in automatisierte Workflows.

Im Kontext kleiner, cloudbasierter Anwendungen bietet Airtable mehrere Vorteile gegenüber klassischen Datenbanksystemen. Die Tabellenstruktur kann ohne aufwendige Deployment- oder Migrationsprozesse angepasst werden, was insbesondere iteratives Arbeiten und prototypische Entwicklungen erleichtert. Darauf hinaus unterstützt die grafische Benutzeroberfläche die manuelle Datenpflege sowie die Fehlersuche. Datensätze sind unmittelbar einsehbar und bearbeitbar, wodurch einfache Auswertungen und Kontrollen direkt in der Anwendung durchgeführt werden können. Demgegenüber stehen typische Einschränkungen gehosteter Plattformen. Dazu zählen unter anderem Begrenzungen hinsichtlich des maximalen Datenvolumens, der Anzahl zulässiger API-Aufrufe sowie die fehlende Unterstützung komplexer Transaktionen. Für klein- bis mittelkomplexe Anwendungen mit moderatem Datenumfang stellt Airtable dennoch eine pragmatische Lösung dar. Es ermöglicht die Realisierung einer zentralen, cloudbasierten Datenhaltung, ohne dass eine eigene Datenbankinfrastruktur betrieben werden muss.

Die verwendete Airtable-Base umfasst vier Tabellen: `User_Sessions` (Dialogzustand), `Customers`, `Rechnungen` und `Rechnungspositionen`.

### 3.3 Google Docs und Google Drive

Google Docs und Google Drive sind zentrale Bestandteile der Google-Workspace-Plattform und bilden gemeinsam eine cloudbasierte Umgebung zur Erstellung, Bearbeitung und Ablage von Dokumenten. Google Docs stellt einen webbasierten Texteditor bereit, der die gleichzeitige Bearbeitung eines Dokuments durch mehrere Personen ermöglicht. Google Drive fungiert als zentraler Dateispeicher und bietet strukturierte Ordner, Freigabefunktionen sowie Versionsverwaltung. Beide Dienste sind vollständig browserbasiert und erlauben einen orts- und geräteunabhängigen Zugriff auf Dokumente, ohne dass eine lokale Office-Software installiert werden muss.

Für automatisierte Systeme ist insbesondere die Programmierschnittstelle von Google Docs von Bedeutung. Sie ermöglicht es, Dokumente nicht nur manuell, sondern auch

programmatisch zu erzeugen und zu verändern. In der Praxis werden hierzu häufig Dokumentvorlagen verwendet, die das Layout sowie feste Textbestandteile definieren. Variable Inhalte werden über Platzhalter integriert, die bei der Dokumentenerstellung durch konkrete Werte aus einem Datensystem ersetzt werden. Auf diese Weise können aus einer einzelnen Vorlage eine Vielzahl individueller Dokumente generiert werden. Typische Anwendungsfälle sind unter anderem die standardisierte Erstellung von Rechnungen, Angeboten oder Serienbriefen, ohne dass jedes Dokument manuell angepasst werden muss [16].

Google Drive ergänzt diese Funktionalität durch eine strukturierte, cloudbasierte Ablage der erzeugten Dokumente. Dateien können in hierarchischen Ordnerstrukturen organisiert, mit Metadaten versehen und über differenzierte Freigabemechanismen geteilt werden. Über die Google-Drive-API lassen sich Ordner und Dateien zudem programmgesteuert anlegen, verschieben, umbenennen oder herunterladen. Dadurch können Dokumentenerzeugung und Archivierung in einen durchgängigen digitalen Prozess integriert werden. Insbesondere für kleinere Organisationen stellt Google Drive damit eine leichtgewichtige Alternative zu klassischen Dokumentenmanagementsystemen dar. Sowohl die manuelle Nutzung über die Weboberfläche als auch automatisierte Workflows auf Basis standardisierter Schnittstellen werden dabei unterstützt [17].

Google Docs erzeugt die Rechnungen auf Basis von Vorlagen, während Google Drive die Dokumente in einer hierarchischen Ordnerstruktur (Jahr/Quartal/Monat/Kunde) ablegt [15].

#### 3.4 Lovable Cloud / Supabase Edge Functions

Für die Webanwendung „ClientHub“ wurde Lovable Cloud als Hosting- und Deployment-Umgebung verwendet. Serverseitige Logik wird dabei nicht über ein klassisches Backend mit dauerhaft laufendem Server umgesetzt, sondern über *Supabase Edge Functions*. Diese Edge Functions sind serverlose HTTP-Endpunkte, die bei Bedarf ausgeführt werden und sich besonders für leichte Integrations- und Proxy-Aufgaben eignen. Im Prototyp dienen sie insbesondere dazu, externe APIs sicher anzubinden, ohne dass API-Schlüssel oder Open Authorization (OAuth)-Geheimnisse im Frontend offen gelegt werden müssen [36].

### 3.4.1 Eingesetzte Edge Functions und Zweck

Im Rahmen des Prototyps wurden zwei Edge Functions umgesetzt, die als Vermittlungsschicht zwischen Frontend und externen Diensten fungieren:

- `airtable-customers`: Bereitstellung von Create, Read, Update, Delete (CRUD)-Operationen zur Abfrage und Verwaltung von Kundendaten in Airtable.
- `google-drive`: Umsetzung des OAuth-2.0-Flows sowie Zugriff auf Ordner- und Dateiinformationen in Google Drive.

### 3.4.2 Aufgerufene APIs

Die Edge Functions kapseln den Zugriff auf mehrere externe Schnittstellen. Für die Datenhaltung wird die Airtable REST-API genutzt (z. B. <https://api.airtable.com/v0/>). Für die Authentifizierung und den Zugriff auf Google Drive wird ein OAuth-2.0-Flow verwendet, der u. a. Google-Endpoints wie <https://accounts.google.com> bzw. <https://oauth2.googleapis.com> einbindet. Der eigentliche Zugriff auf Ordner- und Dateistrukturen erfolgt anschließend über die Google-Drive-API v3 (z. B. <https://googleapis.com/drive/v3/files>).

### 3.4.3 Zentrale Logik in den Edge Functions

Die Implementierung der Edge Functions ist bewusst schlank gehalten und konzentriert sich auf das sichere Weiterreichen von Requests sowie die Normalisierung von Response-Daten.

**Airtable: Kunden abrufen (Mapping)** Die Edge Function `airtable-customers` ruft Datensätze über die Airtable-API ab und transformiert die Antwort in eine frontendfreundliche Struktur, indem `record.id` und `record.fields` zusammengeführt werden:

```
const response = await fetch(airtableUrl, {
  headers: { 'Authorization': `Bearer ${AIRTABLE_API_KEY}` },
});
```

```
const customers = data.records.map((record) => ({
  id: record.id,
  ...record.fields,
}));
```

**Google Drive: OAuth Token Exchange** Die Edge Function `google-drive` führt den Token-Austausch im OAuth-2.0-Prozess serverseitig aus, um Client Secret und weitere vertrauliche Parameter zu schützen:

```
const tokenResponse = await fetch('https://oauth2.googleapis.com/token', {
  method: 'POST',
  body: new URLSearchParams({
    code, client_id, client_secret, redirect_uri,
    grant_type: 'authorization_code',
  }),
});
```

#### 3.4.4 Begründung für Edge Functions

Der Einsatz von Edge Functions bietet im Prototyp mehrere praktische Vorteile:

- **Sicherheit:** API-Keys und OAuth-Geheimnisse verbleiben serverseitig und sind nicht im Browser sichtbar.
- **Cross-Origin Resource Sharing (CORS)-Entkopplung:** Direkte API-Calls aus dem Browser sind häufig durch CORS-Restriktionen limitiert; die Edge Functions fungieren als kontrollierter Proxy.
- **Serverless-Betrieb:** Es ist kein eigener Server und kein klassisches Deployment/Scaling notwendig, da die Edge Functions ereignisbasiert und bedarfsgesteuert ausgeführt werden.
- **Geringe Latenz:** Ausführung am Edge kann zu geringeren Antwortzeiten führen, da Requests geographisch näher am Nutzer verarbeitet werden.

### 3.4.5 Deployment und Konfiguration

Die Edge Functions werden über die Supabase-Konfiguration aktiviert. Im Prototyp wurde die JSON Web Token (JWT)-Verifikation deaktiviert, da die Endpunkte ausschließlich prototypisch und in einer kontrollierten Testumgebung genutzt wurden:

```
[functions.airtable-customers]
verify_jwt = false
```

```
[functions.google-drive]
verify_jwt = false
```

Das Deployment erfolgt in Lovable Cloud automatisch durch Code-Änderungen, sodass kein manuelles Rollout erforderlich ist. Dadurch eignet sich der Ansatz besonders für iterative Prototypenentwicklung.

### 3.4.6 Sicherheitsaspekte

Ein zentraler Grund für die serverseitige Umsetzung ist die sichere Handhabung vertraulicher Zugangsdaten. API-Keys (z. B. Airtable) sowie OAuth-Parameter (Client ID/Client Secret) werden als Secrets hinterlegt und nur innerhalb der Edge Functions verwendet. Zusätzlich können kontrollierte CORS-Header gesetzt werden, um Zugriffe auf definierte Origins zu begrenzen. Im OAuth-Kontext unterstützt der Ansatz außerdem eine saubere Token-Verwaltung (inkl. Refresh-Token-Mechanismus), ohne dass sensitive Daten im Frontend gespeichert werden müssen.

### 3.4.7 Einordnung: Performance im Vergleich zu klassischem Backend

Im Vergleich zu einem traditionellen Backend (dauerhaft laufender Server) sind Edge Functions typischerweise schnell verfügbar und skalieren automatisch pro Request. Während klassische Serverstarts bei Deployment oder Kaltstart-Situationen spürbare Verzögerungen verursachen können, sind serverlose Funktionen in der Regel für kurze, sporadische Integrationsaufgaben geeignet. Gleichzeitig bleiben Grenzen bestehen, z. B.

durch Provider-Limits, Laufzeitbeschränkungen oder die Komplexität umfangreicher Geschäftslogik. Edge Functions kapseln die Zugriffe auf Airtable und Google Drive sicher für das ClientHub-Frontend.

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

Das Frontend der Webanwendung „ClientHub“ wurde als moderne Single-Page-Application umgesetzt. Ziel war es, eine interaktive und übersichtliche Benutzeroberfläche bereitzustellen, die den Zugriff auf Kunden-, Rechnungs- und Dokumentendaten ermöglicht und dabei eng mit den serverseitigen Edge Functions zusammenarbeitet. Die technische Umsetzung basiert auf React in Kombination mit TypeScript sowie einem komponentenbasierten UI- und Styling-Ansatz.

#### 3.5.1 Grundlegende Technologien

Als zentrales Framework kommt React in der Version 18.3.1 in Verbindung mit React DOM 18.3.1 zum Einsatz. React ermöglicht die deklarative Beschreibung von Benutzeroberflächen auf Basis wiederverwendbarer Komponenten und eignet sich insbesondere für datengetriebene Anwendungen mit häufigen Zustandsänderungen. Die Entwicklung erfolgt vollständig in TypeScript, wodurch eine statische Typisierung der Komponenten, Datenmodelle und Schnittstellen erreicht wird. Dies trägt zur besseren Wartbarkeit und zur frühzeitigen Erkennung typischer Fehler bei, etwa bei der Verarbeitung externer API-Daten.

#### 3.5.2 Routing und Zustandsverwaltung

Die Navigation innerhalb der Anwendung wird über clientseitiges Routing realisiert. Hierfür wird `react-router-dom` (Version 6.30.1) eingesetzt, wodurch unterschiedliche Ansichten wie Dashboard, Kundenübersicht oder Dokumentenbrowser als Routen modelliert werden können, ohne dass ein vollständiger Seitenreload erforderlich ist.

Für den Umgang mit serverseitigen Daten wird TanStack Query (`@tanstack/react-query`, Version 5.83.0) verwendet. Diese Bibliothek abstrahiert wiederkehrende Aufgaben wie das Laden, Zwischenspeichern und Aktualisieren von Daten aus externen Quellen.

Im Prototyp werden darüber unter anderem Kundendaten geladen, indem das Frontend eine Supabase Edge Function aufruft und die zurückgegebenen Datensätze als gecachten Server State innerhalb der Anwendung bereitstellt. Fehlerzustände und Ladeindikatoren werden dabei zentral verwaltet und konsistent in der Benutzeroberfläche dargestellt.

### 3.5.3 Formularverarbeitung und Validierung

Zur Umsetzung von Formularen, etwa bei der Anlage neuer Kunden, wird `react-hook-form` (Version 7.61.1) eingesetzt. Die Bibliothek ermöglicht eine performante Verwaltung von Formzuständen und reduziert unnötige Re-Renders. Ergänzend dazu wird `zod` (Version 3.25.76) zur schema-basierten Validierung genutzt. Eingabedaten können dadurch bereits im Frontend auf Vollständigkeit und Plausibilität geprüft werden, bevor sie an die serverseitigen Schnittstellen übermittelt werden.

Die zugrunde liegenden Datenstrukturen werden über TypeScript-Interfaces definiert, wodurch eine klare Trennung zwischen Anzeige-, Formular- und Transportdaten erreicht wird. Ein vereinfachtes Beispiel für ein Kundendatenmodell ist in folgendem Listing dargestellt:

```
export interface Customer {  
    id: string;  
    Firmenname?: string;  
    name?: string;  
    first_name?: string;  
    "E-Mail"?: string;  
    phone?: string;  
    address?: string;  
    city?: string;  
    postal_code?: string;  
    country?: string;  
    ust_id?: string;  
}
```

### 3.5.4 UI-Komponenten und Styling

Für das visuelle Erscheinungsbild der Anwendung wird Tailwind CSS verwendet. Das Styling basiert auf einer stark angepassten Konfiguration mit eigenen Design-Tokens, die über CSS-Variablen definiert sind. Hierzu zählen unter anderem semantische Farbdefinitionen wie „surface“, „success“ oder „warning“ sowie Animationen für Übergänge und Interaktionen. Dieser Ansatz ermöglicht ein konsistentes Erscheinungsbild und erleichtert spätere Anpassungen des Designs.

Als Komponentenbibliothek wird shadcn/ui eingesetzt. Diese stellt vorgefertigte, barriearme UI-Komponenten bereit, die direkt im Projektcode integriert und bei Bedarf angepasst werden können. Im Prototyp kommen unter anderem Layout-Komponenten wie Cards und Tabs, Formularelemente wie Buttons, Inputs und Selects sowie Feedback-Elemente wie Dialoge, Alerts und Toasts zum Einsatz. Die Kombination aus Tailwind CSS und shadcn/ui erlaubt eine flexible Gestaltung bei gleichzeitig hoher Konsistenz der Benutzeroberfläche.

### 3.5.5 Struktur und Datenfluss im Frontend

Die Codebasis des Frontends ist modular aufgebaut und nach fachlichen sowie technischen Aspekten strukturiert. Neben allgemeinen Layout- und UI-Komponenten existieren eigenständige Module für Kundenverwaltung, Dashboard und Dokumentenbrowser. Wiederkehrende Logik, etwa für das Laden von Kundendaten, ist in dedizierten Hooks gekapselt. Der Zugriff auf serverseitige Funktionen erfolgt ausschließlich über den Supabase-Client, der HTTP-Aufrufe an die entsprechenden Edge Functions kapselt und deren Antworten an das Frontend zurückliefert.

### 3.5.6 Responsives Design und Einschränkungen

Das Layout der Anwendung ist teilweise responsiv umgesetzt. Grid-Strukturen und Breakpoints werden genutzt, um Inhalte auf unterschiedlichen Bildschirmgrößen sinnvoll anzurichten. Der Fokus der aktuellen Umsetzung liegt jedoch auf der Desktop-Nutzung; insbesondere die Sidebar-Navigation ist nicht vollständig für mobile Endgeräte optimiert. Diese Einschränkung wurde bewusst in Kauf genommen, da der Prototyp primär für den Einsatz in einer Desktop-orientierten Arbeitsumgebung konzipiert ist.

### 3.5.7 Deployment des Frontends

Das Frontend wird über Lovable Cloud bereitgestellt. Der Build erfolgt als Vite-Produktionsbuild, der anschließend automatisch veröffentlicht wird. Änderungen am Code führen ohne manuelle Eingriffe zu einem neuen Deployment. Die Webanwendung ist sowohl über eine Preview-URL als auch über eine veröffentlichte URL erreichbar. Die enge Verzahnung mit den serverseitigen Edge Functions ermöglicht dabei einen durchgängigen End-to-End-Betrieb der Anwendung. ClientHub verbindet damit das React-Frontend über Edge Functions mit Airtable und Google Drive und stellt eine einheitliche, cloudbasierte Verwaltungsoberfläche für Kunden-, Rechnungs- und Dokumentendaten bereit.

# 4 Konzeption des Systems

Dieses Kapitel beschreibt die Konzeption des zu entwickelnden Systems. Zunächst werden Anforderungen und zentrale Use Cases definiert. Anschließend werden die Zielarchitektur und die Systemübersicht erläutert, gefolgt vom Datenmodell und den Datenflüssen. Darauf aufbauend folgt die detaillierte Beschreibung der Prozessabläufe „Rechnung neu erstellen“ und „Dokument ablegen“. Abschließend wird das Sicherheits- und Datenschutzkonzept vorgestellt.

## 4.1 Anforderungen und Use Cases

Das geplante System richtet sich vor allem an kleine Dienstleistungsunternehmen, Einzelunternehmer sowie kleine und mittlere Gewerbebetriebe. Diese müssen regelmäßig Rechnungen erstellen und verwalten, verfügen jedoch weder über die personellen noch über die finanziellen Ressourcen oder den tatsächlichen Bedarf für umfassende Buchhaltungs- oder ERPs-Systeme. Häufig fehlen zudem vertiefte IT-Kenntnisse, während gleichzeitig nur begrenzte zeitliche Kapazitäten für administrative Aufgaben zur Verfügung stehen. Entsprechend besteht ein Bedarf an einfachen, zuverlässigen und weitgehend automatisierten Prozessen.

Auch Steuerberaterinnen und Steuerberater sind auf vollständig, korrekt und strukturiert archivierte Rechnungsunterlagen angewiesen, um steuerliche Pflichten effizient erfüllen zu können. Studien und Praxisanalysen zeigen, dass manuelle Rechnungsprozesse in kleinen Unternehmen mit erhöhtem Zeitaufwand, einer höheren Fehleranfälligkeit sowie häufigen Medienbrüchen verbunden sind. Dies kann zu einem Mangel an Transparenz in den Abrechnungsprozessen und zu zusätzlichem organisatorischem Aufwand führen [9, 7, 33].

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 fehleranfällig, da Kundendaten und Rechnungspositionen wiederholt neu eingegeben werden müssen. Dadurch steigt das Risiko von Zahlendrehern, unvollständigen Angaben 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 verursacht zusätzlichen Zeit- und Ressourcenaufwand 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 geschäftsrelevanter Dokumente. Um Fehler zu vermeiden und die Effizienz der Arbeitsabläufe zu verbessern, ist es entscheidend, dass alle Daten vollständig, korrekt und einheitlich erfasst werden.

Darüber hinaus ist die Zusammenarbeit mit externen Partnern, insbesondere mit Steuerberaterinnen und Steuerberatern, von großer Bedeutung. Anwender erwarten, dass alle benötigten Unterlagen aktuell, vollständig und gut strukturiert vorliegen, 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 Personen 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 Nutzenden 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 Nutzende 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 im Qualitätsmodell der ISO/IEC 25010 sowie in der Fachliteratur beschrieben sind [20, 34, 21].

Die Use Cases wurden aus den beschriebenen Zielen sowie den daraus resultierenden funktionalen Anforderungen abgeleitet und dienen als Grundlage für die weitere Architektur- und Prozessmodellierung.

### 4.1.1 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 sowie 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.

#### **4.1.2 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 anhand des gewählten Zeitkontexts (Jahr, Quartal, Monat) ein 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.

Diese Use Cases werden durch die cloudbasierte Architektur umgesetzt (siehe Abbildung 4.1).

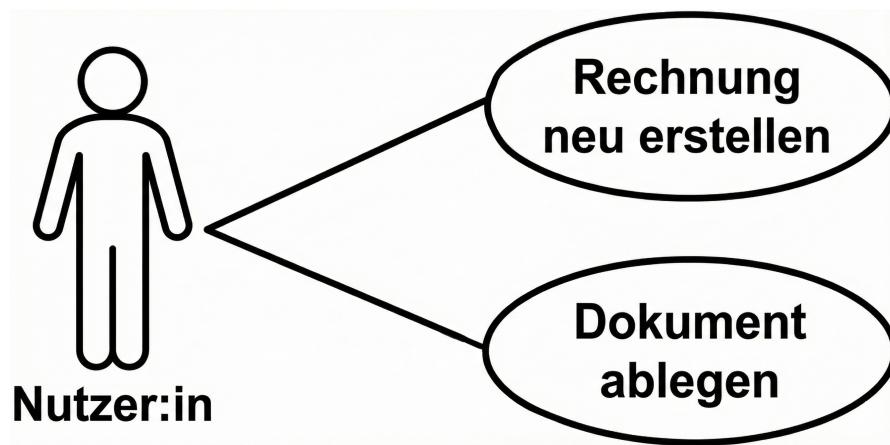


Abbildung 4.1: Use Cases des Systems: Rechnungserstellung und Dokumentenablage.

## 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.

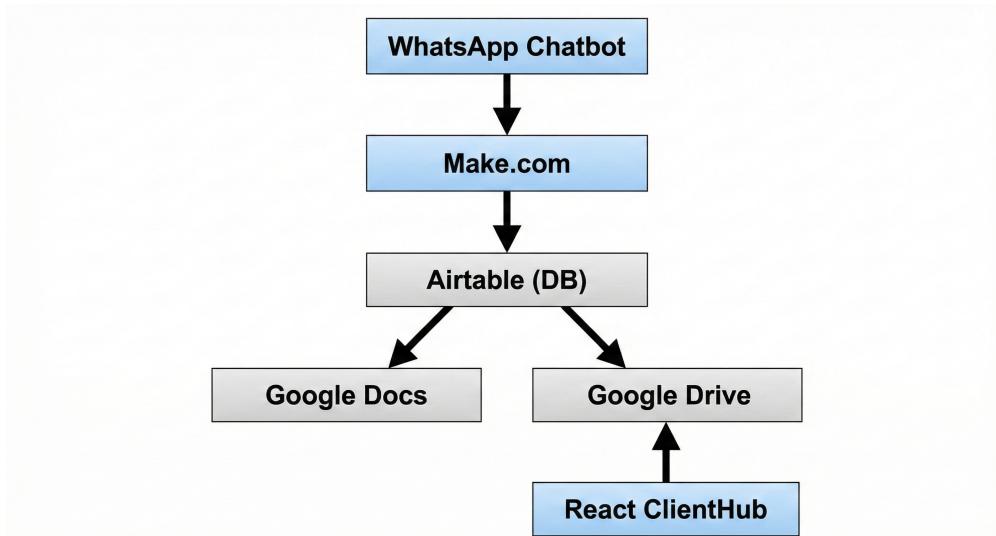


Abbildung 4.2: Zielarchitektur: WhatsApp Chatbot → Make.com → Airtable → Google Docs/Drive → React ClientHub.

Das zentrale Element des Systems ist der über WhatsApp erreichbare Chatbot. Er begleitet die Nutzenden dialoggesteuert durch den Prozess der Rechnungsgenerierung, indem er die einzelnen Schritte strukturiert anleitet. Dies umfasst die Auswahl eines bestehenden Kunden oder die Anlage eines neuen Kundendatensatzes, die Erfassung der Rechnungspositionen sowie die Festlegung von Rechnungsnummer, Leistungszeitraum, Zahlungsfrist und Zahlungsart.

Darüber hinaus stellt der Chatbot die Funktion „Dokument speichern“ bereit. Diese ermöglicht es den Nutzenden, Dateien wie Belege oder bereits erstellte Rechnungen hochzuladen, die anschließend im Dokumentenbereich der Webanwendung „ClientHub“ zur Verfügung stehen.

Eine Integrations- und Workflow-Plattform fungiert als Orchestrierungsschicht und implementiert die Automatisierungslogik des Systems. Sie empfängt Nachrichten vom Chatbot, analysiert den aktuellen Dialogschritt und startet kontextabhängig unterschiedliche externe Dienste [25, 23].

Hierzu zählen insbesondere die Airtable-API zur Erstellung und Aktualisierung von Kunden-, Rechnungs- und Sitzungsdaten (z. B. des aktuellen Prozessschritts) [3], die Google-Docs-API zum Öffnen und Befüllen eines vordefinierten Rechnungsmusters sowie die Google-Drive-API zur Erstellung von Ordner und zum Hochladen von Dateien

[16, 17]. 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 eigenes 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 zeitbezogene 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 Dokumentenbereich der Webanwendung (ClientHub) verfügbar ist.

Die Webanwendung „ClientHub“ ist eine browserbasierte React-Anwendung, die als eigenständiges Modul fungiert und ausschließlich serverseitige Edge Functions nutzt, um auf externe Dienste zuzugreifen. Diese Edge Functions 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 Nutzenden ein Dashboard mit aggregierten Metriken zur Verfügung. Darüber hinaus bietet sie eine Kundenliste mit Such- und Filteroptionen sowie eine Übersicht über die zeitbezogene Ordnerstruktur der Dokumentenablage, 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. [5]. Der konsequente Einsatz wohldefinierter APIs wird als zentraler Baustein integrationsfähiger, verteilter Systeme beschrieben [35].

### 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 [25].

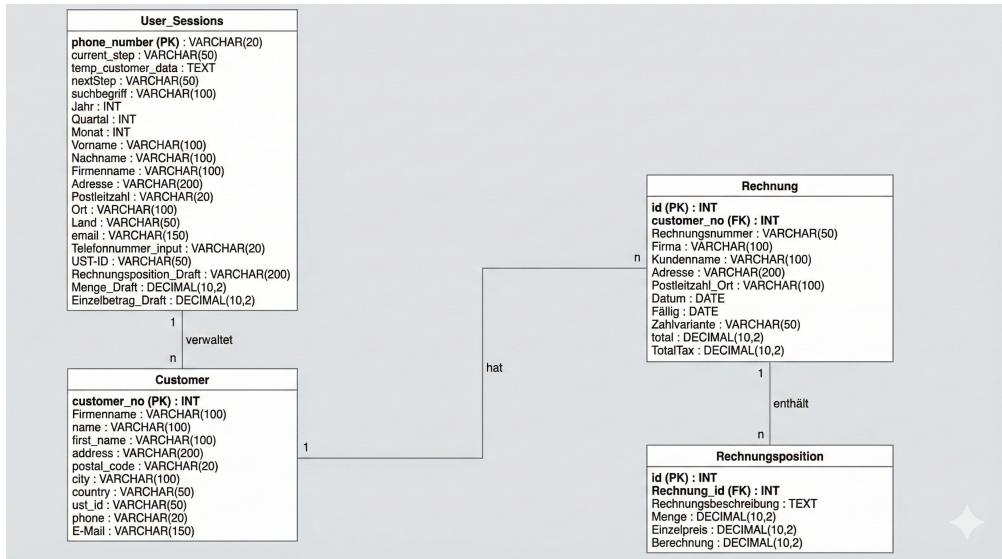


Abbildung 4.3: Airtable-Datenmodell mit Relationen (User\_Sessions, Customers, Rechnungen, Rechnungspositionen).

„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 Datenspeicherschicht, die keine dauerhafte Persistenz besitzt, sondern ausschließlich der Dialogsteuerung und der Vorbereitung der eigentlichen Geschäftsobjekte dient.

Die dauerhaft gültigen Kundendaten werden in der Tabelle „Customers“ 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 „Customers“ überführt.

Bestehende Kunden werden anhand der 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 „Rechnungen“ und „Rechnungspositionen“ abgebildet. Die Tabelle „Rechnungen“ 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 „Customers“, andererseits aus den während des Dialogs in „User\_Sessions“ gespeicherten Eingaben sowie ergänzenden Feldern, die direkt im Chat erfasst und an Airtable übergeben werden.

Zusätzlich werden in „Rechnungen“ aggregierte Beträge gespeichert, indem die Summe aller zugehörigen Rechnungspositionen aus der Tabelle „Rechnungspositionen“ übernommen wird. Diese dient als Grundlage für die Berechnung des Bruttbetrags einschließlich Umsatzsteuer. Die Berechnungen erfolgen über Formel-Felder, die den Nettobetrag mit einem vordefinierten Steuersatz multiplizieren und auf zwei Nachkommastellen runden [4].

Die Tabelle „Rechnungspositionen“ 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 Positionsbeitrags erfolgt über ein Formel-Feld, indem die Menge mit dem numerischen Einzelpreis multipliziert wird. Da eine Rechnung typischerweise mehrere Positionen umfassen kann, wird für jede Position ein separater Datensatz angelegt. Die Zuordnung der einzelnen Rechnungspositionen zu einer Rechnung erfolgt über Referenz- bzw. Verknüpfungsinformationen im Datenmodell und wird durch die Workflow-Logik konsistent sichergestellt.

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 „Customers“, 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 „Rechnungen“ und legt parallel für jede erfasste Rechnungsposition einen Datensatz in „Rechnungspositionen“ an, der die zuvor zwischengespeicherten Leistungsdaten übernimmt. Nach Abschluss der Positionserfassung werden in „Rechnungen“ 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 über die Google-Drive-API als PDF exportiert wird [16, 18]. 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 dialoggesteuert und fokussiert sich 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 für die spätere Ablage notwendigen Metadaten erfasst, insbesondere der zeitliche Kontext (Jahr, Quartal, Monat), und temporär in einer Sitzung gehalten. Auf Basis dieser Parameter stellt der Workflow die entsprechende Ordnerstruktur in Google Drive bereit, sofern diese noch nicht existiert, und lädt das Dokument in den passenden Zeitraumordner hoch.

Der dabei erzeugte Dateilink steht dem System für nachgelagerte Schritte zur Verfügung, etwa zur Anzeige im Dokumentenbereich der Webanwendung. Durch diesen Ablauf wird sichergestellt, dass hochgeladene Dokumente konsistent, nachvollziehbar und eindeutig dem gewählten 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 des Anwendungsfalls „Rechnung neu erstellen“ ist in Abbildung 4.4 dargestellt und wird im Folgenden schrittweise erläutert.

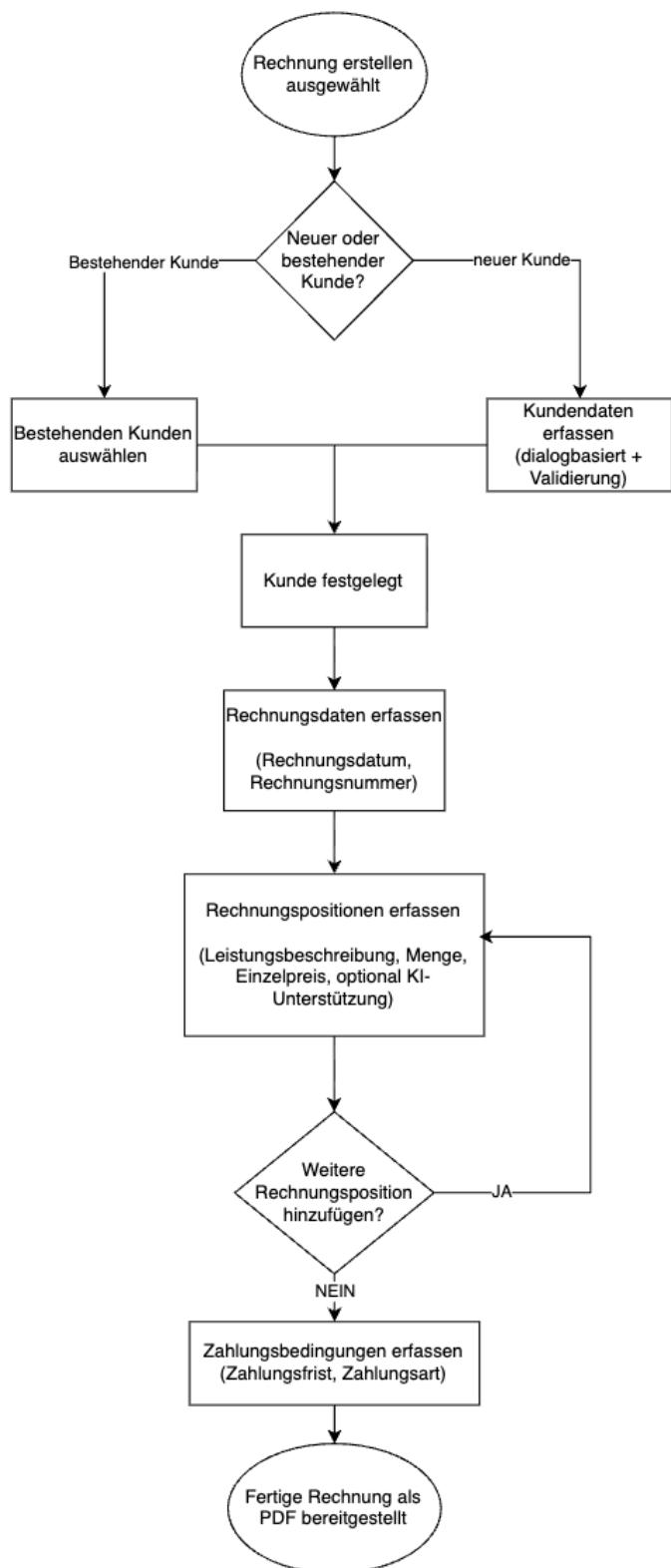


Abbildung 4.4: Prozessablauf des Anwendungsfalls „Rechnung neu erstellen“ vom dialogbasierten Start bis zur Generierung der PDF-Rechnung. 41

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 `Customers` 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 Dialogfade 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 `Rechnungen` 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.YYYY` angeben. Die eingegebene Datumsangabe wird zunächst in `User_Sessions` gespeichert und anschließend in das entsprechende Datumsfeld der Tabelle `Rechnungen` 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 `Rechnungspositionen` 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 `Rechnungen` gespeichert. Darüber hinaus wird die gewünschte Zahlart erfasst, etwa Barzahlung, Kartenzahlung oder Überweisung. Auch diese Information wird als weiteres Attribut in der Tabelle `Rechnungen` hinterlegt. Auf Grundlage der erfassten Daten werden anschließend die relevanten Summen berechnet. Hierzu aggregiert Airtable die Beträge der verknüpften Datensätze aus der Tabelle `Rechnungspositionen` und ergänzt diese um die Umsatzsteuer. Damit liegen alle für die Rechnung erforderlichen Informationen strukturiert in den Tabellen `Customers`, `Rechnungen` und `Rechnungspositionen` 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 `Rechnungen`, `Rechnungspositionen` und `Customers` befüllt. Dabei werden die im Dokument enthaltenen Platzhalter durch die entsprechenden Feldinhalte ersetzt. Das fertig befüllte Dokument wird anschließend als PDF bereitgestellt. 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 Prozess, mit dem Nutzerinnen und Nutzer geschäftsrelevante Dokumente über den WhatsApp-Chatbot hochladen und strukturiert archivieren. Der Ablauf ist in Abbildung 4.5 dargestellt und wird im Folgenden erläutert.

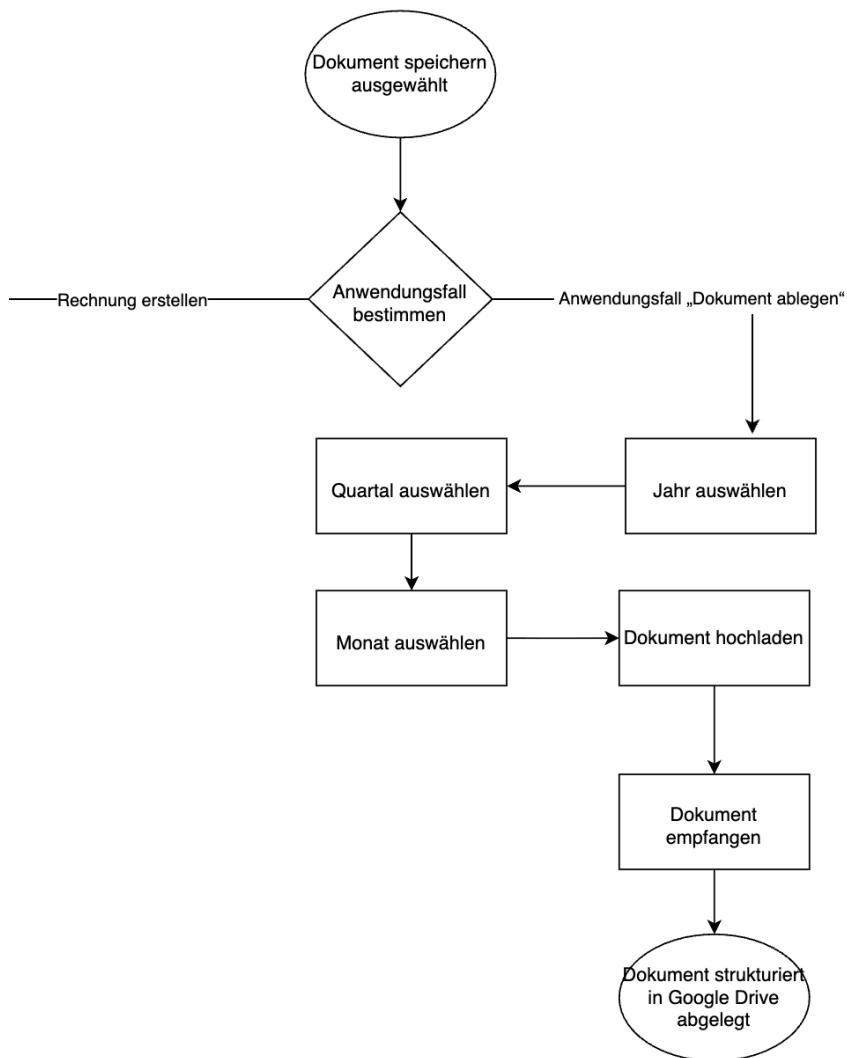


Abbildung 4.5: Prozessablauf des Anwendungsfalls „Dokument ablegen“ vom Chatbot bis zur strukturierten Ablage in Google Drive.

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, wird die 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 [31]. 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 nach Jahr, Quartal und Monat gegliedert ist. Existiert ein Ordner für das ausgewählte Jahr bereits, wird dieser wiederverwendet. Andernfalls wird er neu angelegt. Gleichermaßen 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 vorgeesehenen 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

Die Verarbeitung von Rechnungs- und Kundendaten erfordert besondere Aufmerksamkeit im Hinblick auf Sicherheits- und Datenschutzaspekte, da personenbezogene sowie geschäftlich sensible Informationen verarbeitet werden. Das im Rahmen dieser Arbeit entwickelte System verfolgt daher ein Sicherheits- und Datenschutzkonzept, das sich an den Grundprinzipien der Zweckbindung, Datensparsamkeit und Zugriffsbeschränkung orientiert und diese auf architektonischer Ebene umsetzt.

Im System werden ausschließlich Daten verarbeitet, die für die Erstellung und Ablage von Rechnungen erforderlich sind. Dazu zählen insbesondere Kundenstammdaten wie Name, Adresse und Kontaktdaten sowie rechnungsbezogene Informationen wie Leistungsbeschreibungen, Beträge, Steuersätze und Rechnungsnummern. Zahlungsdaten wie Bankverbindungen oder Kreditkarteninformationen werden bewusst nicht verarbeitet. Dadurch wird der Umfang besonders sensibler Daten reduziert und das Risiko potenzieller Missbrauchsszenarien minimiert.

Die Datenhaltung erfolgt vollständig cloudbasiert. Strukturierte Kunden- und Rechnungsdaten werden zentral in Airtable gespeichert, während die erzeugten Rechnungsdokumente als PDF-Dateien in Google Drive abgelegt werden. Make.com fungiert ausschließlich als Integrations- und Orchestrierungsschicht und speichert keine Daten dauerhaft, sondern verarbeitet Informationen lediglich transient innerhalb der definierten Szenarien. Die Web-Anwendung „ClientHub“ greift über serverseitige Edge Functions auf diese Dienste zu und hält selbst keine persistenten Daten vor.

Ein zentrales Element des Sicherheitskonzepts ist die konsequente Trennung von Frontend und sicherheitskritischer Integrationslogik, wie in Abschnitt 3.4 beschrieben. Der Zugriff auf externe APIs wie Airtable und Google Drive erfolgt nicht direkt aus dem Browser, sondern ausschließlich über serverseitige Supabase Edge Functions. API-Schlüssel sowie OAuth-Zugangsdaten werden als Secrets in der Serverumgebung hinterlegt und sind für das Frontend nicht einsehbar. Dieses Vorgehen reduziert das Risiko der Offenlegung sensibler Zugangsdaten und stellt sicher, dass sicherheitsrelevante Operationen kontrolliert und nachvollziehbar durchgeführt werden.

Die Authentifizierung gegenüber Google Drive erfolgt über den OAuth-2.0-Standard, der eine zeitlich begrenzte und widerrufbare Autorisierung ermöglicht. Sämtliche Datenübertragungen zwischen den Systemkomponenten erfolgen über HTTPS-verschlüsselte Verbindungen. Ergänzend können Zugriffe auf die Edge Functions durch gezielte CORS-Regeln eingeschränkt werden, um ausschließlich definierte Ursprünge zuzulassen. Die Umsetzung dieser Maßnahmen ist Bestandteil der in Kapitel 5 beschriebenen Implementierung.

Aus datenschutzrechtlicher Sicht orientiert sich das System an den grundlegenden Anforderungen der Datenschutz-Grundverordnung (DSGVO). Die Verarbeitung der Daten erfolgt ausschließlich zweckgebunden zur Unterstützung der Rechnungs- und Dokumentenverwaltung. Es findet keine Profilbildung, keine automatisierte Entscheidungsfindung und keine Weitergabe der Daten an unbeteiligte Dritte statt. Durch die Nutzung etablierter Cloud-Dienste wird auf Anbieter zurückgegriffen, die gängige Sicherheitsstandards und Compliance-Anforderungen erfüllen [12, 37]. Da es sich um einen prototypischen Ansatz handelt, erfolgt der Einsatz in einer kontrollierten Umgebung und nicht im produktiven Masseneinsatz.

Gleichzeitig bestehen systembedingte Grenzen und Risiken. Die Abhängigkeit von externen Cloud-Anbietern bedeutet, dass Aspekte wie Datenstandort, Verfügbarkeit und langfristige Speicherung nicht vollständig unter eigener Kontrolle stehen. Zudem ist der Einsatz eines Messengers als Eingabekanal aus datenschutzrechtlicher Sicht kritisch zu betrachten, da Teile der Kommunikation außerhalb des eigentlichen Systems verarbeitet werden. Für einen produktiven Einsatz wären daher weiterführende Maßnahmen wie feinere Rollen- und Rechtekonzepte, eine detaillierte Protokollierung sowie vertragliche Regelungen zur Auftragsverarbeitung erforderlich.

Insgesamt zeigt das Sicherheits- und Datenschutzkonzept, dass auch mit vergleichsweise einfachen, cloudbasierten Bausteinen ein verantwortungsvoller Umgang mit sensiblen Da-

ten möglich ist, sofern klare architektonische Trennungen, serverseitige Zugriffskontrollen und eine bewusste Begrenzung des Datenumfangs umgesetzt werden.

Die Konzeption mit WhatsApp-Chatbot, Make.com-Workflows, Airtable-Datenmodell und ClientHub-Frontend ist technisch umsetzbar (siehe Kapitel 5).

# 5 Implementierung

Dieses Kapitel beschreibt die technische Umsetzung des in Kapitel 4 konzipierten Systems. Zunächst wird die Implementierung des Chatbots über Make.com-Szenarien erläutert. Anschließend folgt die Datenhaltung mit Airtable und die Web-Anwendung ClientHub. Darauf aufbauend werden automatische Dokumentenerstellung in Google Docs, Ablagestrategie in Google Drive sowie die ClientHub WebApp mit Frontend, Edge Functions und API-Anbindungen detailliert beschrieben.

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

Das in Kapitel 4 konzipierte Chatbot-System wurde mithilfe der Automatisierungsplattform **Make.com** umgesetzt. Die Dialoglogik basiert auf mehreren Szenarien, die über Webhooks Ereignisse auslösen, eingehende Nachrichten verarbeiten und Daten zwischen WhatsApp, Airtable, Google Docs und Google Drive austauschen.

Die Grundlage der Kommunikation bildet die **WhatsApp Business Cloud API**. Eingehende Nachrichten werden über ein *Watch Events*-Modul in Make als Trigger erfasst [25]. Jedes empfangene Ereignis initiiert einen eindeutigen Ablauf, wodurch eine synchrone Verarbeitung der Nachrichten sichergestellt wird.

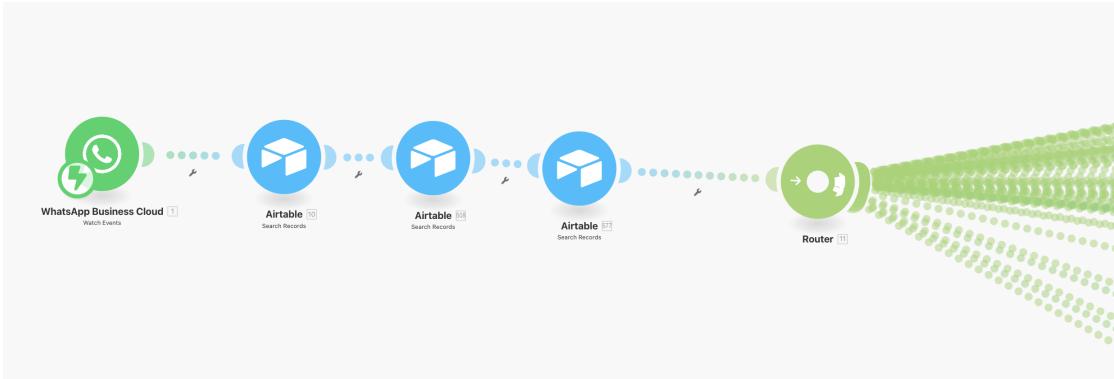


Abbildung 5.1: Hauptszenario des Chatbots in Make.com: WhatsApp-Trigger → Router → Airtable/Google-Integration → Antwort.

### Szenario-Start und Nutzerregistrierung

Nach der Aktivierung durch ein eingehendes WhatsApp-Ereignis ruft das Szenario die zugehörige Telefonnummer ab und prüft in Airtable, ob bereits ein Nutzerstatus existiert. Hierzu werden über das Modul *Search records* der Airtable-App Datensätze in der Tabelle `User_Sessions` abgefragt [27].

Wird kein entsprechender Eintrag gefunden, legt das System automatisch eine neue Sitzung an, speichert die Telefonnummer im Feld `phone_number` und setzt den Wert von `current_step` auf *menu*. Im Anschluss wird dem Nutzer eine automatisierte Willkommensnachricht übermittelt, die dazu auffordert, durch die Eingabe von "Start" den Prozess zu beginnen. Der Versand der Nachricht erfolgt über das Modul *Send a message* der WhatsApp-Integration [25].

### Dialogsteuerung und Routing

Die gesamte Interaktionslogik wird durch einen zentralen Router gesteuert, der anhand des Felds `current_step` in der Tabelle `User_Sessions` entscheidet, welcher Teilablauf aktiviert wird. Dieses Vorgehen ermöglicht eine zustandsbasierte Steuerung des Dialogs.

Sendet ein Nutzer beispielsweise den Befehl "Start", wird der Menü-Zustand (*menu*) aktiviert. In diesem Schritt erhält der Nutzer eine Nachricht mit zwei interaktiven Buttons: "Rechnung erstellen" und "Dokument speichern". Bei Auswahl eines Buttons aktualisiert

das Szenario das Statusfeld `current_step` in Airtable und verzweigt anschließend in den entsprechenden Ablauf.

Zur Verarbeitung mehrerer möglicher Folgeschritte kommen in Make Router-Module in Kombination mit Filterbedingungen zum Einsatz. Diese aktivieren abhängig von Zustands- und Eingabewerten unterschiedliche Ausgänge und steuern so den weiteren Ablauf des Szenarios [23].

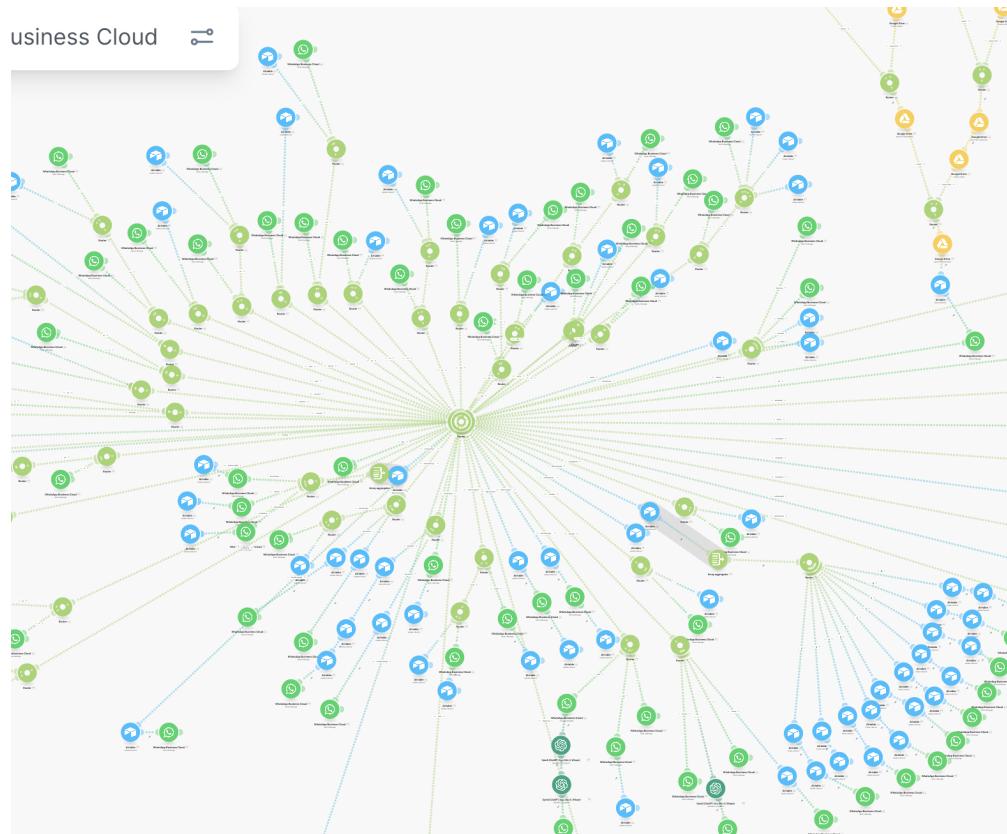


Abbildung 5.2: Router-Modul mit `current_step`-Steuerung: Verzweigung nach Dialogzustand.

### Ablauf “Rechnung erstellen”

Im Pfad “**Rechnung erstellen**” beginnt die Datenerfassung mit der Kundenauswahl. Der Nutzer kann entscheiden, ob ein neuer Kunde angelegt oder ein bestehender Kunde gesucht werden soll.

Bei Auswahl der Option *Kunde suchen* fordert der Chatbot einen Suchbegriff an, speichert diesen in der Tabelle `User_Sessions` und durchsucht anschließend die Airtable-Tabelle `Customers` nach passenden Einträgen. Die Suchabfrage basiert auf einer OR-Kombination mehrerer Felder, darunter Name, Firmenname und Adresse, und verwendet zusätzlich die Funktion `LOWER()` zur case-insensitiven Suche [27].

Die ermittelten Ergebnisse werden mithilfe eines *Array aggregators* zu einem Array zusammengeführt. Dessen Länge wird in nachgelagerten Filtermodulen ausgewertet, um zu unterscheiden, ob kein, ein oder mehrere Treffer vorliegen [23].

- **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 `Customers` gespeichert und zugleich als Relation in `Rechnungen` 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 `Rechnungen` und `Rechnungspositionen` 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 [28]. Damit ist der End-to-End-Prozess der Rechnungserstellung abgeschlossen.

### Ablauf “Dokument speichern”

Wird im Hauptmenü statt der Rechnungsfunktion die Option *Dokument speichern* gewählt, wird der zweite zentrale Pfad zur strukturierten **Dokumentenablage** aktiviert. Der Chatbot fragt Jahr, Quartal und Monat sequenziell ab, speichert die jeweiligen Angaben in der Tabelle `User_Sessions` und fordert den Nutzer anschließend zum Hochladen des Dokuments auf. Auf diese Weise wird der zeitliche Kontext der Ablage eindeutig definiert und durch den Nutzer bestätigt.

Im nächsten Schritt prüft das Szenario mithilfe des Moduls *Search files/folders*, ob im Google Drive bereits entsprechende Ordnerstrukturen existieren. Falls dies nicht der Fall ist, werden die erforderlichen Verzeichnisse automatisch in der Reihenfolge Telefonnummer → Jahr → Quartal → Monat angelegt. Nach erfolgreicher Prüfung oder Erstellung lädt das Modul *Upload a file* das Dokument in den vorgesehenen Ordner hoch und speichert den erzeugten Drive-Link in Airtable. Dadurch kann das Dokument später über die Web-Anwendung konsistent identifiziert und abgerufen werden [29].

## 5.2 Datenhaltung und Web-Anwendung (Airtable und ClientHub)

Die im Chatbot erfassten Informationen werden in einer zentralen Airtable-Base gespeichert, die als „Single Source of Truth“ für alle kunden- und rechnungsbezogenen Daten fungiert [22]. Auf diese Weise greifen sowohl das Chatbot-Szenario in Make als auch die Web-Anwendung (ClientHub) auf denselben konsistenten Datenbestand zu.

Beide Systeme können dadurch fachliche Objekte wie Kunden oder Rechnungen unabhängig voneinander lesen und bearbeiten, ohne dass redundante Datenhaltungen entstehen [2].

### 5.2.1 Tabellenstruktur und Datenmodell in Airtable

Das operative Datenmodell gliedert sich in vier zentrale Tabellen: `User_Sessions`, `Customers`, `Rechnungen` und `Rechnungspositionen`. Die Tabelle `User_Sessions` dient als technischer Sitzungs- und Zwischenspeicher für den Chatbot.

#	phon...	curre...	temp...	nextS...	such...	Jahr	Quartal	Monat	Vorn...	Nach...	Firme...	Adre...	Postl...	Ort	Land	email	Telef...	UST-ID	Rech...	Menge	Einze...
1	4916...	Einzel...	rechn...	kunde...					Max	Muste...	Muste...	Muste...	2211	Hamb...	Deuts...	Muste...	+4917...	DE13...	Bestic...	15	13€

Abbildung 5.3: `User_Sessions`-Tabelle mit `current_step`-Steuerung und Telefonnummern.

Sie enthält unter anderem die Felder `phone_number`, `current_step`, `temp_customer_data` und `nextStep` sowie weitere Attribute zur Speicherung temporärer Eingaben, etwa Suchbegriffe, Zeitangaben (Jahr, Quartal, Monat) und positionsbezogene Daten wie Rechnungsposition, Menge und Einzelbetrag. Dadurch wird eine zustandsbasierte Dialogführung ermöglicht, ohne dass Nutzer ihre Angaben mehrfach wiederholen müssen [22].

Im Rahmen der Kundenerstellung werden vorläufige Stammdaten wie Vorname, Nachname, Firmenname, Adresse, Postleitzahl, Ort, Land, E-Mail-Adresse, Telefonnummer und USt-ID zunächst in `User_Sessions` gespeichert. Erst nach einer expliziten Bestätigung werden diese Daten in die Tabelle `Customers` überführt.

Die Tabelle `Customers` bildet die fachliche Kundenstammdatenbasis und enthält Felder wie `customer_no`, Firmenname, Name, Adresse sowie Kontakt- und Identifikationsdaten. Diese Struktur ermöglicht sowohl eine eindeutige Identifikation über die Kundennummer als auch eine flexible, textbasierte Suche über mehrere Attribute, wie sie im Chatbot mithilfe von `filterByFormula`-Ausdrücken sowie Funktionen wie `LOWER` und  `FIND` realisiert wird [4].

Die Tabelle `Rechnungen` speichert kopfbezogene Rechnungsdaten wie Kundendaten, Rechnungsdatum, Rechnungsnummer, Fälligkeit, Zahlvariante sowie Gesamt- und Steuerbeträge. Die zugehörigen Einzelpositionen werden in der Tabelle `Rechnungspositionen` abgelegt. Diese enthalten neben den zur Referenzierung notwendigen Kopffeldern positionsspezifische Informationen wie Leistungsbeschreibung, Menge, Einzelpreis und Steueranteile. Die Kopplung über die Rechnungsnummer erlaubt es dem Web-Frontend, sowohl aggregierte Summen als auch einzelne Rechnungspositionen gezielt abzufragen.

### 5.2.2 Interaktion zwischen Chatbot und Datenhaltung

Das Chatbot-Szenario in Make greift über die Airtable-Module „Search records“, „Create a record“ und „Update a record“ auf die beschriebenen Tabellen zu [22, 27]. Bei jeder eingehenden Nachricht wird zunächst der aktuelle Sitzungsdatensatz in `User_Sessions` gesucht oder neu angelegt. Dabei werden die Felder `phone_number` und `current_step` gesetzt, um den Nutzer eindeutig zu identifizieren und den aktuellen Dialogstatus zu speichern.

Im Verlauf der Interaktion speichert der Chatbot Zwischenstände wie Suchbegriffe, ausgewählte Kunden, Rechnungsdatum, Rechnungsnummer, Rechnungspositionen, Mengen und Einzelbeträge in der Tabelle `User_Sessions`. Nach finaler Bestätigung durch den Nutzer werden diese Daten in die fachlichen Tabellen `Customers`, `Rechnungen` und `Rechnungspositionen` überführt. Auf diese Weise bleibt die in Abschnitt 5.1 beschriebene zustandsbasierte Logik eng mit der Datenhaltung verknüpft, ohne dass zusätzliche serverseitige Komponenten erforderlich sind.

Die Datenstruktur unterstützt sowohl lesende als auch schreibende Zugriffe aus verschiedenen Kanälen. Während der Chatbot primär auf die Erstellung und Aktualisierung von Sitzungsdaten, Kunden, Rechnungen und Rechnungspositionen fokussiert ist, nutzt die Web-Anwendung denselben Datenbestand hauptsächlich zur Anzeige, Filterung und Auswertung.

Airtable stellt hierfür eine HTTP-basierte REST-API bereit, die sowohl in Make als auch in serverseitigen Komponenten über API-Keys und `filterByFormula`-Parameter verwendet wird, um gezielt Teilmengen der Daten abzurufen [2]. Dadurch wird eine klare Trennung zwischen Dialoglogik und fachlicher Persistenz erreicht, da sämtliche Fachdaten ausschließlich in Airtable verwaltet werden.

### 5.2.3 Architektur der Web-Anwendung (ClientHub)

Die Web-Anwendung „ClientHub“ ist als React-Frontend realisiert und greift nicht direkt auf Airtable oder Google Drive zu. Stattdessen werden Supabase Edge Functions als serverseitige Mittelschicht eingesetzt. Das Frontend ruft diese Funktionen über `supabase.functions.invoke()` auf und übergibt die jeweils benötigten Parameter, beispielsweise zur Kundensuche, zur Abfrage von Rechnungslisten oder zur Auflistung von Dokumenten in einem bestimmten Drive-Ordner [36].

Die Business-Logik zur Kommunikation mit der Airtable-API und der Google-Drive-API ist dabei in zwei Edge Functions gekapselt. Die Funktion `airtable-customers` übernimmt kundenbezogene Datenzugriffe, während `google-drive` für Datei- und Ordneroperationen zuständig ist.

Supabase Edge Functions werden in einer serverlosen, global verteilten Infrastruktur ausgeführt und eignen sich insbesondere für HTTP-basierte Integrationen mit Drittsystemen [36]. Innerhalb der Funktionen werden notwendige API-Schlüssel und Zugriffstoken als Secrets verwaltet, sodass sie weder im Browser noch im Frontend-Code sichtbar sind.

Die Edge Functions nehmen HTTP-Anfragen des Frontends entgegen, führen Authentifizierungs- und Autorisierungsprüfungen durch und rufen anschließend die entsprechenden Airtable- oder Google-Drive-Endpunkte auf. Die Ergebnisse werden in einem für das Frontend geeigneten JSON-Format zurückgegeben [36]. Dadurch wird die Angriffsfläche reduziert und die Einhaltung von Sicherheitsanforderungen, etwa der Schutz von API-Schlüsseln und der Zugriff ausschließlich durch authentifizierte Nutzer, erleichtert.

### 5.2.4 Funktionale Abgrenzung zwischen Web-App und Chatbot

Web-Anwendung und Chatbot adressieren unterschiedliche Nutzungsszenarien, greifen jedoch auf denselben Datenbestand in Airtable zu. Über den Chatbot können Nutzer insbesondere Rechnungen dialogbasiert erstellen und Dokumente hochladen, während die Web-Anwendung eine übersichtliche, tabellarische und navigierbare Darstellung der Daten bereitstellt.

In der Web-App stehen unter anderem Funktionen wie eine Kundenübersicht in Tabelleform, die Anlage neuer Kunden über Formular-Dialoge, Direktlinks für E-Mail- und Telefonkontakt sowie ein interaktiver Ordnerbaum zur Navigation in Google Drive zur Verfügung. Ergänzt wird dies durch Ein-Klick-Downloads von Dateien und Dashboard-Ansichten mit aggregierten Kennzahlen [17].



Abbildung 5.4: ClientHub Web-App: Kundenübersicht mit Such-/Filterfunktion.

Diese Funktionen unterstützen insbesondere Recherche-, Verwaltungs- und Analyseaufgaben im Büro- und Desktop-Kontext und ergänzen damit die dialogbasierte Interaktion des Chatbots.

Der Chatbot eignet sich hingegen besonders für mobile Nutzungsszenarien oder Situationen, in denen Informationen unmittelbar im Gesprächsfluss erfasst werden sollen, etwa direkt nach einer erbrachten Dienstleistung. Da beide Kanäle auf die gemeinsamen Tabellen Customers, Rechnungen und Rechnungspositionen zugreifen, bleiben Stamm- und Belegdaten konsistent, unabhängig davon, ob sie über den WhatsApp-Dialog oder die Web-Anwendung angelegt oder verändert werden [2]. Die Kombination aus dialogbasierter Erfassung und webbasierter Verwaltung erhöht dadurch sowohl die Datenqualität als auch die Flexibilität der Nutzung.

### 5.3 Automatische Dokumentenerstellung in Google Docs

Nach Abschluss des Chat-Dialogs, in dem sämtliche für die Rechnungserstellung erforderlichen Informationen erfasst wurden, erzeugt das System automatisiert ein formatiertes Rechnungsdokument auf Grundlage einer Google-Docs-Vorlage [28]. Anschließend wird das Dokument programmgesteuert als PDF exportiert und dem Nutzer unmittelbar im WhatsApp-Chat bereitgestellt, sodass keine manuelle Nachbearbeitung in einem Textverarbeitungsprogramm erforderlich ist [16].

### 5.3.1 Rechnungsvorlage und Platzhalter

Die Grundlage der automatisierten Dokumentenerstellung bildet ein vordefiniertes Google-Docs-Dokument, das das Layout der Rechnung festlegt, einschließlich Briefkopf, Absenderinformationen, Tabellenstruktur für die Rechnungspositionen sowie der Zusammenfassung der Beträge [14]. In dieses Template sind Platzhalter in geschweiften Klammern integriert, beispielsweise `{ {company_name} }`, `{ {company_companyName} }`, `{ {company_address} }`, `{ {company_date} }` und `{ {company_invoiceNr} }`, die zur Laufzeit durch konkrete Werte aus den Airtable-Tabellen `Customers` und `Rechnungen` ersetzt werden.

Für die Abbildung der Rechnungspositionen werden analoge Platzhalter wie `{ {items_beschreibung} }`, `{ {{items_menge} } }`, `{ {{items_preis} } }` und `{ {{items_total} } }` verwendet. Zusätzlich kommen Felder für Zwischensummen und Gesamtbezüge (`{ {{company_subtotal} } }`, `{ {{company_total} } }`) zum Einsatz, sodass sowohl Einzelpositionen als auch aggregierte Werte dynamisch befüllt werden können [14].

Das zugrunde liegende Platzhalterkonzept entspricht dem in der Google-Docs-API beschriebenen Mail-Merge-Ansatz, bei dem Textmarken in einem Dokument automatisiert durch Daten aus externen Systemen ersetzt werden. Auf diese Weise lassen sich aus einem einheitlichen Template eine Vielzahl individueller Rechnungsdokumente erzeugen [14].

### 5.3.2 Generierung und Versand der PDF-Rechnung

Sobald der Nutzer im Chatbot alle Angaben zu Kunde, Rechnungsdatum, Rechnungsnummer, Rechnungspositionen, Zahlungsfrist und Zahlungsart bestätigt hat, wird der automatische Generierungsprozess ausgelöst. Zunächst werden die relevanten Datensätze aus den Airtable-Tabellen `Customers`, `Rechnungen` und `Rechnungspositionen` ausgelesen und so aufbereitet, dass für jeden im Dokument definierten Platzhalter ein eindeutiger Wert vorliegt [22]. Anschließend wird über die Google-Docs-Integration ein neues Dokument als Kopie der Rechnungsvorlage erzeugt, wobei die im Template enthaltenen Platzhalter mit den übergebenen Daten ersetzt werden. Auf diese Weise entsteht ein vollständig ausgefülltes Rechnungsdokument [24].

Im nächsten Schritt wird das erzeugte Dokument über die Google-Docs- beziehungsweise Google-Drive-Schnittstelle serverseitig in das PDF-Format konvertiert und als Datei

bereitgestellt [16, 13]. Die resultierende PDF-Rechnung wird in der aktuellen Implementierung nicht automatisch in einer strukturierten Ordnerhierarchie in Google Drive abgelegt, sondern primär für den unmittelbaren Versand über den WhatsApp-Chat verwendet. Hierzu wird die PDF-Datei über das WhatsApp-Sendemodul an den Nutzer übertragen, der das Dokument lokal speichern oder weiterleiten kann. Eine optionale, langfristige Ablage in Google Drive erfolgt erst in einem separaten Prozessschritt über den Menüpunkt „Dokument speichern“ im Chatbot [26, 17].

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

Die Ablage von Dokumenten in Google Drive erfolgt nach einer klaren, zeitbasierten Ordnerstruktur, um Belege auch bei wachsendem Datenvolumen schnell auffindbar und revisionssicher zu verwalten. Alle vom Nutzer hochgeladenen Dokumente werden im Pfad „Dokument speichern“ des Chatbots systematisch nach Jahr, Quartal und Monat eingesortiert.

Wählt der Nutzer im Hauptmenü den Pfad „Dokument speichern“, fragt der Chatbot nacheinander Jahr, Quartal und Monat ab und speichert die Antworten in der Tabelle `User_Sessions` (`Jahr`, `Quartal`, `Monat`). Dadurch ist zum Zeitpunkt des Datei-Upserts bereits festgelegt, in welchem Zeitraum das Dokument archiviert werden soll, was insbesondere für buchhalterische Auswertungen und steuerliche Nachweise vorteilhaft ist. Erst danach wird der Nutzer aufgefordert, das betreffende Dokument (z. B. eine Rechnung im PDF-Format) zu übermitteln.

Nach Eingang der Datei prüft das Szenario schrittweise die Google-Drive-Ordnerhierarchie (Telefonnummer → Jahr → Quartal → Monat) und legt fehlende Ebenen automatisch an [15]. Der kundenbezogene Stammordner (Telefonnummer) wird identifiziert bzw. erstellt, gefolgt von den zeitbasierten Unterordnern für Jahr, Quartal und Monat nach demselben iterativen Muster.

Auf diese Weise entsteht eine konsistente Pfadstruktur der Form:

/Kunden/<Telefonnummer>/<Jahr>/<Quartal>/<Monat>

ohne dass bereits bestehende Ordner dupliziert werden. Ist der Zielordner vollständig bestimmt, wird die vom Nutzer gesendete Datei in genau diesen Ordner hochgeladen

und der resultierende Drive-Link in Airtable gespeichert, sodass die Web-Anwendung die Dokumente später kontextbezogen anzeigen und beispielsweise nach Jahr oder Quartal filtern kann [15].

Die Ablagestrategie ist bewusst vom Prozess der automatischen Rechnungserzeugung getrennt. Wie in Abschnitt 5.3 beschrieben, wird die Rechnung zunächst als PDF generiert und direkt im WhatsApp-Chat bereitgestellt, ohne automatisch in einem Drive-Ordner archiviert zu werden [13]. Möchte der Nutzer die Rechnung zusätzlich in der Jahres-/Quartals-/Monatsstruktur ablegen, ruft er erneut den Pfad „Dokument speichern“ auf und lädt die PDF-Datei hoch; das System ordnet sie dann anhand der gewählten Zeitparameter dem passenden Ordner zu.

### 5.5 Implementierung der ClientHub WebApp

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

ClientHub ist eine React 18.3.1 Single-Page-Application (Vite/TypeScript), die für Desktop-Nutzung am Arbeitsplatz konzipiert ist. Die Sidebar-Navigation (280 px) umfasst die Bereiche Dashboard, Kunden, Ordnerbaum und Connections.

##### Views:

- DashboardView: StatsCards (Kunden/Dokumente).
- CustomerTable: CRUD-Tabelle (Prototyp: Read/Create).
- FolderBrowser: Rekursiver Drive-Baum ( $\leq 6$  Ebenen).
- ConnectionsView: OAuth-Setup.

##### Stack:

## 5 Implementierung

Tabelle 5.1: ClientHub Frontend

Bibliothek	Zweck
TanStack Query 5.83.0	Fetching/Caching/Sync
react-hook-form 7.61	Form-State
zod 3.25.76	Schema-Validierung
shadcn/ui	Table/Dialog/Card
Tailwind CSS	Design
lucide-react	Icons

React Query: Auto-Refresh nach Mutations. zod: Pflichtfelder (Firmenname, E-Mail).

The screenshot shows the ClientHub dashboard interface. On the left, a sidebar lists 'Dashboard', 'Kunden', 'Kundenportal', and 'Verbindungen'. A message at the bottom says 'Verbunden mit Airtable & Google Drive'. The main area has a 'Dashboard' header with stats: 'Gesamt Kunden 5 +2 diesen Monat', 'Aktive Kunden 3 60% aktiv', 'Kundenordner 5', and 'Dokumente 12 +4 diese Woche'. Below this is a 'Kundenliste' section showing 2 customers from Airtable with columns: Firmenname, Vorname, Nachname, and E-Mail. One entry is 'Muster GmbH' with 'Max Mustermann' and 'Muster.mustermann@hotmail.de'. To the right is a 'Google Drive' section titled 'Kundenportal Ordner' with a tree view of files. It shows a folder '491633931258' containing '2024/Quartal 2/Mai/Bildschirmfoto\_2025-09-19...', '2024/Quartal 3', '2025/Quartal 1', and '2025/Quartal 2'. Other files include 'Rechnung 0001'. The interface uses a dark theme with blue highlights for active sections.

Abbildung 5.5: ClientHub Dashboard mit Kunden- & Drive-Übersicht.

### 5.5.2 Edge Functions in Lovable Cloud

Zwei Edge Functions auf Basis von Deno Runtime (Deno) und TypeScript (TS) kapseln sicherheitskritische Logik und schützen dabei insbesondere APIs-Zugangsdaten (Airtable und Google Drive).

Tabelle 5.2: Edge Functions

Name	Funktionen
airtable-customers	GET/POST (Base: appN3KaFLqeV4FURN)
google-drive	OAuth, Folders/Tree (v3)

Config: `verify_jwt=false` (Prototyp). Secrets: Keys/OAuth.

```
[functions.airtable-customers]
verify_jwt = false
```

Deployment: Git-Push (Preview: id-preview-7485e714...lovable.app).

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

Frontend → Edge → APIs (CORS/Secrets serverseitig).

**Airtable:** Table `tbljfpEqd4bbEYUqB` (Firmenname, E-Mail, Adresse). Vollständige Liste (Prototyp, kein serverseitiges Filter).

**Google Drive:** Authorization Code Flow:

- Frontend: Authorization URL → Code empfangen.
- Edge: Code + Secret → Token (`oauth2.googleapis.com/token`).
- Drive API: `drive/v3/files` (Ordner/Dateien).

Beispiel:

```
supabase.functions.invoke("airtable-customers", { method: "GET" });
```

Da Chatbot und Web-Anwendung auf dieselbe Airtable-Base zugreifen, bleiben Kunden- und Belegdaten kanalübergreifend konsistent. Die technische Umsetzung bestätigt die Machbarkeit der Konzeption (Kapitel 4). Die Evaluation der Implementierung folgt in Kapitel 6.

# 6 Evaluation

Dieses Kapitel beschreibt die systematische Überprüfung des entwickelten Prototyps. Zunächst wird das Testkonzept und die Testumgebung erläutert. Anschließend folgen die Ergebnisse der funktionalen Tests und der Performance-Analyse. Darauf aufbauend werden manueller und automatisierter Prozess verglichen sowie die Usability bewertet. Abschließend werden Grenzen des Systems und Risiken diskutiert.

## 6.1 Testkonzept und Testumgebung

Ziel des Testkonzepts war es, die grundsätzliche Funktionsfähigkeit und Machbarkeit des entwickelten Prototyps zu überprüfen. Im Vordergrund stand die manuelle, funktionale Validierung der End-to-End-Prozesse für die beiden zentralen Use Cases „Rechnung erstellen“ und „Dokument speichern“. Die Tests sollten zeigen, dass die konzipierten Automatisierungen technisch lauffähig sind und Nutzereingaben zu erwarteten Ergebnissen führen. Neben der reinen Funktionsfähigkeit wurden Datenkonsistenz, korrekte Dokumentgenerierung, grundlegende Stabilität und das subjektive Reaktionsverhalten des Systems betrachtet.

Der Testumfang umfasste alle wesentlichen Komponenten der prototypischen Lösung: WhatsApp-Chatbot mit Orchestrationsworkflows, End-to-End-Prozess zur Rechnungserstellung sowie Dokumentenablage in der strukturierten Ordnerhierarchie. Darüber hinaus wurden die Anzeige- und Abruffunktionen der Webanwendung „ClientHub“ über serverseitige Backend-Funktionen einbezogen. Nicht Gegenstand waren Sicherheits-, Penetrations- tests, Last-/Stresstests bei hoher Nutzerzahl oder API-Auslastung sowie automatisierte Unit-/Integrationstests, da der Fokus auf funktionalem Prototyp lag.

Die Tests wurden in einer einfachen Entwicklungs- und Testumgebung durchgeführt, die der Zielarchitektur entspricht, jedoch ohne produktive Nutzung. Für die Webanwendung kam eine separate Entwicklungs-/Testversion zum Einsatz, die ausschließlich der Anzeige und Prüfung der gespeicherten Daten diente. Automatisierungen wurden direkt

in der Integrationsplattform manuell gestartet, ohne Trennung in Test- und Produktiv-Workspaces. Datenhaltung erfolgte in Tabellen mit Testdaten, Dokumentenablage in separaten Testordnern des Cloud-Speichers. Für Chats wurde ein eigener Test-Account verwendet.

Als Testdaten kamen ausschließlich synthetische, manuell angelegte Datensätze zum Einsatz, um reale Personen- oder Firmendaten zu vermeiden. Es wurden Beispielkunden mit typischen Konstellationen erstellt: Privatkunden ohne Umsatzsteuer-ID sowie Firmenkunden mit Firmenname und optionaler Umsatzsteuer-ID. Darauf basierend entstanden Beispielrechnungen mit 1–3 Positionen, um wesentliche Varianten abzudecken. Fehlerhafte Eingaben wurden situativ getestet (leere/falsche Chat-Eingaben, Dialogabbrüche), um die Stabilität des Dialogflusses zu prüfen.

Das Testvorgehen war bewusst manuell, explorativ und end-to-end ausgerichtet. Tests erfolgten parallel zur Entwicklung: Automatisierungen manuell starten → vollständige Chat-Dialoge durchlaufen → Ergebnisse in Datenhaltung, Dokumentenablage und Weboberfläche prüfen. Formale Testfälle oder automatisierte Tests entfielen; Abweichungen wurden direkt in Workflows korrigiert und Abläufe wiederholt. Jeder zentrale Use Case wurde 2–5× getestet, um reproduzierbare Funktionsfähigkeit zu sichern.

Ein Test galt als bestanden, wenn der End-to-End-Prozess ohne Abbruch durchlief, das erwartete Ergebnis sichtbar erzeugt wurde und der Ablauf bei Wiederholung stabil blieb. Dazu zählten: korrekt generierte PDF-Rechnung, richtige Dokumentenablage, vollständige Datenspeicherung in Tabellen und Anzeige in der Webanwendung. Kleinere Fehler wurden iterativ behoben: Workflows angepasst, erneut getestet, verbleibende Limitationen dokumentiert.

## 6.2 Funktionale Tests

Die funktionalen Tests zielten darauf ab, die zentralen Use Cases des Systems manuell und end-to-end zu überprüfen. Im Mittelpunkt standen Rechnungserstellung, Dokumentenablage und Anzeige der gespeicherten Daten in der Webanwendung. Die Tests wurden über die vorgesehenen Benutzeroberflächen gestartet und bis zur Erzeugung der erwarteten Artefakte (PDF-Dokumente, Datensätze, Ordnerstrukturen) vollständig durchlaufen.

Der Use-Case „Rechnung mit bestehendem Kunden und einer Position erstellen“ startete im Chatbot mit der Auswahl der Funktion „Rechnung erstellen“, der Auswahl eines

vorhandenen Kunden und der Eingabe einer einzelnen Rechnungsposition. Erwartet wurde, dass in der Datenhaltung neue Einträge in den Tabellen für Rechnung und Rechnungsposition angelegt und mit dem bestehenden Kundendatensatz verknüpft werden und der Nutzer eine korrekt befüllte PDF-Rechnung im Chat erhält. Nach kleineren Korrekturen an der Datenuordnung lief dieser Ablauf stabil und reproduzierbar. Ein zweiter Use-Case betrachtete die Erstellung einer Rechnung für einen neu anzulegenden Kunden mit mehreren Positionen. Hierbei wurden im Dialog zunächst die Kundendaten erfasst, anschließend mehrere Positionen eingegeben und schließlich eine vollständige PDF-Rechnung erzeugt. Erwartet wurden ein neuer Kundendatensatz sowie konsistent verknüpfte Rechnungs- und Positionsdatensätze; nach Anpassung der Dialoglogik zur Verarbeitung mehrerer Positionen funktionierte dieser Ablauf ebenfalls zuverlässig.

Der Use-Case „Dokument speichern“ überprüfte die Ablage beliebiger Dateien anhand einer zuvor im Chat ausgewählten Struktur aus Jahr, Quartal und Monat. Aus Sicht der Funktionalität war gefordert, dass die Ordnerhierarchie im Speicherdiensst automatisch erzeugt oder wiederverwendet wird, die Datei im passenden Zielordner abgelegt und ein Verweis auf das Dokument in der Datenhaltung gespeichert wird. Nach Korrektur einzelner fehlerhafter Pfadzuordnungen wurden diese Anforderungen erfüllt. Ein weiterer Testfall betraf die Anzeige von Rechnungen und Dokumenten in der Webanwendung „ClientHub“. Über die Weboberfläche wurden Kunden ausgewählt und die zugehörigen Rechnungen und Dokumente angezeigt; dabei sollten die in der Datenhaltung gespeicherten Informationen konsistent abgebildet und verlinkte PDF-Dokumente direkt geöffnet oder heruntergeladen werden können. Die Tests bestätigten, dass die Lesefunktionen der Web-App zuverlässig arbeiten, wobei der Fokus auf der Anzeige und nicht auf der Bearbeitung lag.

Neben den regulären Abläufen wurden auch Fehler- und Sonderfälle betrachtet. Im Use-Case „Fehlerfall – ungültige Eingabe im Chat“ wurden etwa leere Eingaben, Texte anstelle erwarteter Zahlen und der Abbruch eines Dialogs in der Mitte getestet. Ziel war es, sicherzustellen, dass der Prozess nicht unkontrolliert abbricht, sondern der Dialog im aktuellen oder vorherigen Schritt verbleibt und der Nutzer zur erneuten Eingabe aufgefordert wird. Die Tests zeigten, dass einfache Eingabefehler durch die Dialoglogik und Zustandsverwaltung abgefangen werden und der Nutzer in vielen Fällen sinnvoll weitergeführt wird; komplexere Sonderfälle werden im aktuellen Prototyp jedoch noch nicht in allen Varianten vollständig validiert und können in Einzelfällen zu suboptimalen Zuständen führen.

Insgesamt lässt sich festhalten, dass die zentralen End-to-End-Prozesse funktional stabil und reproduzierbar arbeiten. Die Kernfunktionen zur Rechnungserstellung, zur strukturierten Ablage von Dokumenten und zur Anzeige der Daten im ClientHub sind umgesetzt und liefern die erwarteten Ergebnisse. Vollständig erfüllt sind damit insbesondere die Anforderungen an die grundlegende Funktionsfähigkeit und Datenkonsistenz der Hauptprozesse. Nur teilweise erfüllt sind hingegen eine umfassende Fehlerbehandlung aller Rand- und Sonderfälle sowie eine formale, testfallbasierte Dokumentation, die für einen prototypischen Entwicklungsstand jedoch nicht im Fokus stand.

### **6.3 Performance-Analyse**

Ziel der Performance-Analyse war nicht, formale Antwortzeiten im Millisekundenbereich zu bestimmen, sondern zu überprüfen, ob die End-to-End-Prozesse für einzelne Nutzer vollständig durchführbar sind und ob die dabei entstehenden Wartezeiten subjektiv noch nachvollziehbar und akzeptabel erscheinen. Im Vordergrund stand damit eine praxisnahe Einschätzung der tatsächlichen Wartezeit im Chat sowie beim Dokumenten-Upload, nicht die technische Optimierung einzelner Schnittstellen oder Komponenten. Die Analyse betrachtet daher exemplarische Abläufe und bewertet deren Verhalten qualitativ aus Nutzersicht.

Untersucht wurden drei typische Szenarien. Im ersten Szenario wurde die Rechnungserstellung für einen neuen Kunden mit einer Rechnungsposition betrachtet: Vom Start der Funktion „Rechnung erstellen“ im Chat bis zur Rückgabe der fertigen PDF-Rechnung lagen die beobachteten Durchlaufzeiten typischerweise im Bereich von drei bis vier Minuten. Ein zweites Szenario betrachtete die Rechnungserstellung für einen bereits bestehenden Kunden; hier verkürzte sich die Dauer auf etwa zwei bis drei Minuten, da der Schritt der Kundenerfassung entfiel. Im dritten Szenario wurde der Prozess „Dokument speichern“ gemessen, bei dem der Nutzer Jahr, Quartal und Monat auswählt und anschließend eine Datei sendet. Die Zeit vom Start der Funktion bis zur Ablage der Datei im vorgesehenen Ordner lag hier bei etwa ein bis zwei Minuten. Für die Webanwendung „ClientHub“ wurde keine formale Zeitmessung durchgeführt, da die Anzeige- und Klickvorgänge subjektiv sehr schnell ablaufen und keinen dominierenden Einfluss auf den Gesamtprozess haben.

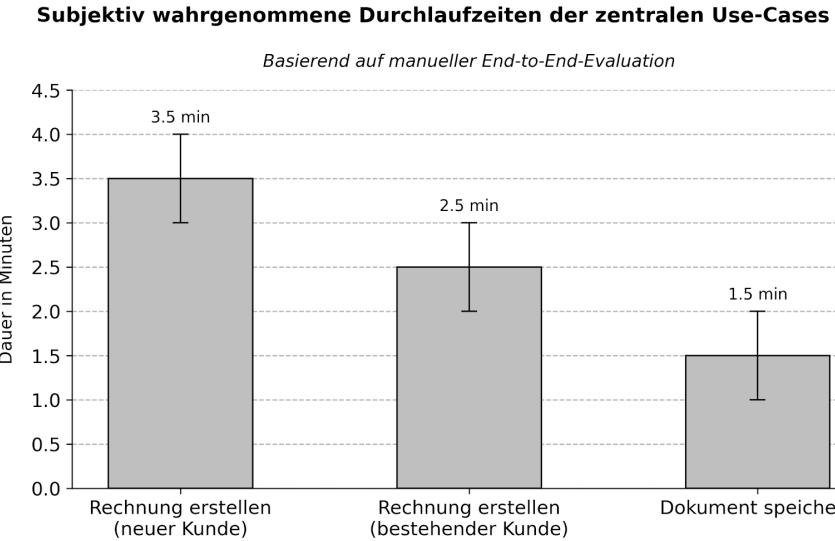


Abbildung 6.1: Subjektiv wahrgenommene Durchlaufzeiten der zentralen Use-Cases.

Die Messungen erfolgten bewusst einfach, indem die Dauer mit Hilfe einer Uhr beziehungsweise groben Zeitabschätzung aus Sicht des Nutzers erfasst wurde. Pro Szenario wurden mehrere Durchläufe durchgeführt, insbesondere nach Anpassungen der Automatisierungen, um die Reproduzierbarkeit der beobachteten Zeiten zu prüfen. Technische Timing-Werkzeuge wie Browser-Entwicklertools oder detaillierte Logauswertungen kamen nicht zum Einsatz, da der Schwerpunkt auf der wahrgenommenen Performance in einer typischen Nutzungssituation lag.

Die beobachteten Zeiten wurden im Wesentlichen durch die Ausführung der Automatisierungsworflows und die Antwortzeiten des Messaging-Kanals geprägt. Insbesondere die sequentielle Abarbeitung mehrerer Module in der Integrationsplattform, interne Wartezeiten zwischen einzelnen Schritten sowie das Auffinden und Setzen des jeweils korrekten Dialogpfads im Chat tragen zur Gesamtdauer bei. Andere Faktoren wie die Performance der Webanwendung oder die lokale Internetverbindung spielten im Vergleich eine untergeordnete Rolle. Auffällige Timeouts oder vollständige Abbrüche traten im Rahmen der Tests nicht auf; die Abläufe waren zwar teilweise langsam, wurden jedoch stets vollständig beendet, sodass keine explizite Timeout-Analyse erforderlich war.

Aus Nutzersicht zeigt sich ein differenziertes Bild. Grundsätzlich sind alle zentralen Funktionen nutzbar, da die Prozesse technisch korrekt durchlaufen und die erwarteten Ergebnisse – insbesondere Rechnungs-PDFs und abgelegte Dokumente – erzeugt werden.

Gleichzeitig wird deutlich, dass der Use-Case „Rechnung erstellen“ durch die Vielzahl einzelner Dialogschritte, insbesondere bei der Neuanlage eines Kunden, als vergleichsweise langwierig wahrgenommen werden kann. Die schrittweise Abfrage von Vorname, Nachname, Adresse, Postleitzahl usw. erhöht zwar die Datenqualität, verlängert aber den Prozess und birgt das Risiko, dass Nutzer den Vorgang vorzeitig abbrechen. Demgegenüber wirkt der Use-Case „Dokument speichern“ deutlich effizienter, da nur wenige Rückfragen erforderlich sind und das Ergebnis schnell sichtbar wird. Insgesamt bestätigt die Analyse den grundsätzlichen Nutzen des Automatisierungsansatzes, macht aber gleichzeitig deutlich, dass aus Sicht der Nutzererfahrung insbesondere bei der Rechnungserstellung Optimierungspotenzial besteht, etwa durch die Reduktion sequentieller Schritte oder die weitere Straffung der Dialogführung.

## 6.4 Vergleich: manueller vs. automatisierter Prozess

Bei der betrachteten Zielgruppe, also kleinen Unternehmen und Selbstständigen ohne spezialisiertes Buchhaltungs- oder ERP-System, erfolgt die Rechnungserstellung typischerweise in einem weitgehend manuellen Ablauf. Zunächst werden die benötigten Informationen wie Kundendaten, Leistungsbeschreibung und Preise gesammelt, anschließend wird eine Word- oder Excel-Vorlage geöffnet und die Daten werden manuell eingetragen und formatiert. Die Vergabe der Rechnungsnummer, der Export als PDF sowie das Speichern und Versenden der Datei, etwa per E-Mail oder Messenger, erfolgen ebenfalls manuell. Dieser Ansatz ist mit einem hohen Zeitaufwand verbunden, führt leicht zu Eingabe- und Formatierungsfehlern und resultiert häufig in einer unstrukturierten Ablage, wodurch Rechnungen später nur schwer auffindbar sind [38].

Das entwickelte System unterstützt und vereinfacht diesen Prozess, ohne ihn vollständig zu automatisieren. Durch die Nutzung eines einheitlichen Templates entfällt die manuelle Formatierung, und die PDF-Rechnung wird nach Abschluss der Datenerfassung automatisch erzeugt und an den Nutzer zurückgesendet. Rechnungsdaten und Dokumente werden zentral gespeichert, und die Datenerfassung erfolgt strukturiert über einen dialogbasierten Chat. Nicht automatisiert ist hingegen die Vergabe der Rechnungsnummer: Sie wird weiterhin vom Nutzer manuell eingegeben, ohne automatische Prüfung auf doppelte oder ungültige Nummern. Gleichzeitig entstehen neue Aufwände, da die Eingabe im Chat in mehrere Einzelschritte zerlegt ist; insbesondere bei der Neuanlage von Kunden werden

zahlreiche Felder wie Name, Adresse, Postleitzahl und Ort nacheinander abgefragt, was den Prozess verlängert und als ermüdend empfunden werden kann.

Auch bei der Dokumentenablage zeigt sich ein deutlicher Unterschied zwischen manueller und automatisierter Vorgehensweise. In der Ausgangssituation werden Dateien gespeichert oder heruntergeladen, ein vermeintlich passender Ordner wird manuell gesucht oder neu angelegt und der Dateiname frei vergeben. Späteres Wiederfinden erfolgt über manuelle Suche, was durch uneinheitliche Ordner- und Dateinamen sowie fehlende zeitliche oder inhaltliche Struktur erschwert wird [9, 7]. Im entwickelten System werden Dokumente dagegen automatisch in einer fest definierten Ordnerstruktur nach Jahr, Quartal und Monat abgelegt; der zugehörige Link wird in der Datenhaltung gespeichert, sodass ein schneller Zugriff über Chat oder Webanwendung möglich ist. Der manuelle Aufwand bei der Ablage reduziert sich damit deutlich, auch wenn der Nutzer den Prozess weiterhin aktiv über die Funktion „Dokument speichern“ starten und die Datei selbst bereitstellen muss.

Im Hinblick auf Zeit- und Aufwandsersparnis ergibt sich ein gemischtes Bild. Die manuelle Erstellung einer Rechnung nimmt, abhängig von Erfahrung und Vorlagenqualität, typischerweise etwa fünf bis fünfzehn Minuten in Anspruch. Mit dem Prototyp reduziert sich die Dauer bei bestehenden Kunden auf etwa zwei bis drei Minuten, bei neuen Kunden auf etwa drei bis vier Minuten, wobei die manuelle Eingabe der Rechnungsnummer enthalten ist. Die Dokumentenablage verkürzt sich von etwa zwei bis fünf Minuten bei rein manueller Vorgehensweise auf etwa ein bis zwei Minuten im automatisierten Prozess. Insgesamt verringert das System damit den Gesamtaufwand, insbesondere bei wiederkehrenden Vorgängen und bei der Ablage.

Hinsichtlich Fehleranfälligkeit und Transparenz bietet der automatisierte Ansatz ebenfalls Vorteile. Durch die zentrale Nutzung eines Templates werden Formatierungsfehler reduziert, und die schrittweise, strukturierte Datenerfassung senkt das Risiko vergessener Pflichtfelder. Gleichzeitig bleiben Fehler bei der Rechnungsnummer möglich, da diese weiterhin manuell vergeben wird und keine automatisierte Prüfung erfolgt. Positiv wirkt sich die zentrale Datenhaltung aus, die eine klare Verknüpfung zwischen Kunden, Rechnungen und Dokumenten ermöglicht und durch die einheitliche Ordnerstruktur die Nachvollziehbarkeit erhöht. Insgesamt zeigt der Vergleich, dass der Prototyp insbesondere bei Dokumentenerzeugung, Ablage und Datenstrukturierung einen klaren Mehrwert bietet, während die Rechnungserstellung zwar funktional nutzbar ist, aber durch lange Dialoge

und verbleibende manuelle Schritte noch Optimierungspotenzial für einen produktiven Einsatz aufweist.

## 6.5 Usability-Bewertung

Die Usability wurde subjektiv nach ISO 9241-11 bewertet [1]. Die Bewertung erfolgte als heuristische Selbsteinschätzung des Entwicklers auf Basis der Kriterien Effektivität, Effizienz und Zufriedenheit, da keine formale Nutzerstudie durchgeführt wurde.

Kriterium	Note	Begründung
Effektivität	4/5	Kern-Use-Cases vollständig (Abschnitt 6.2)
Effizienz	3/5	Prozessdauer ca. 2–4 Minuten (Abschnitt 6.3)
Zuverlässigkeit	3/5	Einfache Fehler werden korrekt behandelt (Abschnitt 6.2)
Lernbarkeit	4/5	Intuitive Nutzung durch Chat-Interaktion (vgl. Abbildung 5.4)
Zufriedenheit	4/5	Natürliche Dialoge und positive Nutzererfahrung
<b>Gesamt</b>	<b>3.6/5</b>	<b>Nutzbar mit Optimierungspotenzial</b>

Die Ergebnisse zeigen, dass der Prototyp insgesamt gut nutzbar ist, jedoch noch Verbesserungspotenzial aufweist. Insbesondere die Effektivität und Lernbarkeit werden positiv bewertet, da die zentralen Use-Cases vollständig umgesetzt sind und der dialogbasierte Chat eine intuitive Nutzung ermöglicht. Auch die subjektive Zufriedenheit ist hoch, da die Interaktion als natürlich und verständlich wahrgenommen wird.

Demgegenüber fallen Effizienz und Zuverlässigkeit etwas schwächer aus. Die vergleichsweise langen Durchlaufzeiten einzelner Prozesse sowie die noch nicht vollständig abgedeckte Fehlerbehandlung in Randfällen wirken sich negativ auf die Bewertung aus. Insgesamt bestätigt die Usability-Bewertung, dass der Prototyp für den praktischen Einsatz grundsätzlich geeignet ist, zugleich aber insbesondere hinsichtlich Prozessdauer, Dialogführung und Fehlerrobustheit weiter optimiert werden sollte.

## 6.6 Grenzen des Systems und Risiken

Der entwickelte Prototyp weist mehrere technische und funktionale Grenzen auf, die insbesondere im Hinblick auf einen produktiven Einsatz berücksichtigt werden müssen. Diese Einschränkungen resultieren sowohl aus dem prototypischen Charakter der Umsetzung als auch aus bewussten Designentscheidungen im Rahmen der Arbeit.

Eine zentrale Grenze betrifft die Effizienz der dialogbasierten Interaktion. Die schrittweise Erfassung von Kundendaten, bei der einzelne Informationen wie Name, Adresse, Postleitzahl und Ort nacheinander abgefragt werden, führt insbesondere bei der Neuanlage von Kunden zu vergleichsweise langen Prozesszeiten von etwa drei bis vier Minuten (vgl. Abschnitt 6.3). Für zeitlich stark eingebundene Selbstständige kann dieser Ablauf als zu lang empfunden werden und die Bereitschaft zur vollständigen Durchführung des Prozesses verringern.

Darüber hinaus ist die Fehlerbehandlung im aktuellen Prototyp nur eingeschränkt umgesetzt. Während einfache Eingabefehler im Chat abgefangen werden, werden komplexere Sonderfälle, wie ungültige oder doppelt vergebene Rechnungsnummern sowie abgebrochene Dialoge an ungünstigen Stellen, nicht in allen Varianten zuverlässig behandelt (vgl. Abschnitt 6.2). Dies kann in Einzelfällen zu inkonsistenten Zuständen oder unvollständigen Datensätzen führen.

Ein weiteres Risiko ergibt sich aus den eingesetzten Technologien und deren Abhängigkeiten. Der Prototyp basiert auf mehreren externen Cloud-Diensten, darunter die Integrationsplattform Make.com, die Datenhaltung in Airtable sowie verschiedene Google-APIs. Dadurch entsteht ein hohes Maß an Vendor Lock-in. Ausfälle oder Änderungen dieser Dienste können unmittelbar zu einem Ausfall oder einer Einschränkung der Systemfunktionalität führen.

Auch die Skalierbarkeit des Systems ist begrenzt. Es wurden keine Last- oder Stresstests durchgeführt, und die genutzten Dienste unterliegen technischen Einschränkungen, etwa durch API-Limits wie die Begrenzung der Anfragefrequenz bei Airtable. Bei gleichzeitiger Nutzung durch mehrere Anwender könnten diese Limits zu Verzögerungen oder Fehlern führen, wodurch der Prototyp derzeit nur eingeschränkt für einen Mehrnutzerbetrieb geeignet ist.

Zusätzlich bestehen Risiken im Bereich Datenschutz und Compliance. Personenbezogene Daten wie Telefonnummern werden in der Datenhaltung gespeichert, ohne dass Löschfris-

ten, Zugriffsbeschränkungen oder eine detaillierte Protokollierung implementiert wurden. Auch ein umfassendes Backup- oder Wiederherstellungskonzept für die in Airtable und im Cloud-Speicher abgelegten Daten ist nicht vorgesehen. Für einen produktiven Einsatz, insbesondere unter Berücksichtigung der Datenschutz-Grundverordnung (DSGVO), wären hier weitergehende Maßnahmen erforderlich.

Zusammenfassend lässt sich festhalten, dass der entwickelte Prototyp für die Einzelnutzung und zur Demonstration des grundsätzlichen Ansatzes geeignet ist. Ein produktiver Einsatz in einem Mehrnutzerkontext würde jedoch ein technisches und organisatorisches Hardening des Systems erfordern, insbesondere in Bezug auf Fehlerrobustheit, Skalierbarkeit, Datenschutz und Betriebssicherheit. Die Evaluation bestätigt die technische Machbarkeit des Konzepts. Die identifizierten Optimierungspotenziale fließen in den Ausblick (Kapitel 7.4).

Trotz der genannten Einschränkungen zeigt die Evaluation, dass der entwickelte Prototyp die konzeptionellen Ziele erfüllt und die technische Umsetzbarkeit eines dialogbasierten Ansatzes für administrative Prozesse erfolgreich demonstriert.

# 7 Fazit und Ausblick

## 7.1 Zusammenfassung der Ergebnisse

Ziel dieser Arbeit war es, einen niedrigschwlligen Ansatz zur teilautomatisierten Rechnungs- und Dokumentenverarbeitung für kleine Dienstleistungsunternehmen und Selbstständige zu entwickeln. Auf Basis der analysierten Ausgangssituation wurde ein prototypisches System entworfen, das einen Chatbot zur dialogbasierten Datenerfassung mit einer cloudbasierten Datenhaltung und einer ergänzenden Web-Anwendung zur Einsicht und Verwaltung von Kunden, Rechnungen und Dokumenten kombiniert. Im Fokus stand dabei die Unterstützung realer Arbeitsabläufe ohne Einführung komplexer Buchhaltungssoftware und mit möglichst geringem Anpassungsaufwand für die Nutzenden.

Im Rahmen der Konzeption wurden zentrale Use-Cases definiert, insbesondere die Erstellung von Rechnungen sowie die strukturierte Ablage administrativer Dokumente. Darauf aufbauend wurde eine Zielarchitektur entwickelt, die eine Integrationsplattform zur Orchestrierung der Prozesse, einen Messaging-Kanal für die Chatbot-Interaktion, ein cloudbasiertes Datenmodell für Kunden- und Rechnungsdaten sowie eine Web-Oberfläche für den Überblick und den Zugriff auf die erzeugten Dokumente umfasst. Die Dialogführung im Chatbot und die Struktur der zugrunde liegenden Daten wurden so gestaltet, dass eine schrittweise und nachvollziehbare Erfassung der relevanten Informationen ermöglicht wird.

Die prototypische Implementierung zeigt, dass sich mit vergleichsweise einfachen, standardisierten Cloud-Diensten ein durchgängiger End-to-End-Prozess realisieren lässt. Rechnungsdaten können dialoggeführt erfasst, in einer zentralen Datenbasis gespeichert und auf Basis eines Templates automatisiert zu PDF-Rechnungen generiert werden. Gleichzeitig werden Rechnungen und weitere Dokumente in einer konsistenten, nach Zeiträumen strukturierten Ordnerhierarchie abgelegt und über eine Web-Anwendung auffindbar gemacht.

Die Evaluation anhand exemplarischer Szenarien ergab, dass die Kernfunktionen zur Rechnungserstellung, zur Ablage von Dokumenten und zur Anzeige der Daten grundsätzlich stabil ablaufen und die erwarteten Ergebnisse liefern.

Gleichzeitig wurden im Rahmen der Untersuchung auch Grenzen und Herausforderungen des prototypischen Ansatzes deutlich. Dazu zählen unter anderem der vergleichsweise lange Prozess der Rechnungserstellung bei der Neuanlage von Kunden aufgrund vieler einzelner Dialogschritte, eine nur teilweise abgedeckte Fehler- und Sonderfallbehandlung sowie eine Performance, die aus Nutzersicht zwar akzeptabel, aber nicht in allen Fällen optimal ist. Zudem zeigt sich, dass die Nutzung externer Cloud- und Integrationsdienste zwar die Entwicklung erleichtert, jedoch Abhängigkeiten und potenzielle Einschränkungen hinsichtlich Skalierung, Kosten und langfristiger Wartbarkeit mit sich bringt. Insgesamt belegen die Ergebnisse jedoch, dass der gewählte Ansatz grundsätzlich geeignet ist, Medienbrüche zu reduzieren und wiederkehrende manuelle Tätigkeiten in der Rechnungs- und Dokumentenverarbeitung zu unterstützen.

## 7.2 Beantwortung der Forschungsfragen

In diesem Abschnitt werden die in Abschnitt 1.3 formulierten Forschungsfragen auf Basis der in dieser Arbeit erzielten Ergebnisse beantwortet.

### Forschungsfrage 1

*Wie lässt sich ein prototypisches System zur Erstellung und strukturierten Ablage von Rechnungen und administrativen Dokumenten für kleine Dienstleistungsunternehmen und Selbstständige auf Basis gängiger cloudbasierter Werkzeuge konzipieren?*

Die Arbeit zeigt, dass ein solches System durch die Kombination eines Chatbots, einer Integrationsplattform, einer cloudbasierten Datenhaltung und einer ergänzenden Web-Anwendung realisiert werden kann. Die entwickelte Zielarchitektur nutzt einen bestehenden Kommunikationskanal für die Interaktion, eine zentrale Datenbasis für Kunden- und Rechnungsinformationen sowie standardisierte Cloud-Dienste für die automatisierte Dokumentenerzeugung und strukturierte Ablage. Damit konnte ein durchgängiger, auf wiederkehrende administrative Abläufe zugeschnittener End-to-End-Prozess konzipiert werden.

## Forschungsfrage 2

*Inwieweit kann ein Chatbot, der an einen bestehenden Kommunikationskanal angebunden ist, die dialogbasierte Erfassung von Rechnungs- und Dokumentendaten unterstützen, sodass Medienbrüche reduziert und wiederkehrende manuelle Arbeitsschritte verringert werden?*

Die prototypische Umsetzung zeigt, dass ein Chatbot die Erfassung von Rechnungs- und Dokumentendaten grundsätzlich wirksam unterstützen kann, indem er Nutzende Schritt für Schritt durch den Prozess führt und die erfassten Informationen direkt an die nachgelagerten Automatisierungen weitergibt. Medienbrüche werden reduziert, da relevante Daten nicht mehr in separaten Office-Dokumenten oder E-Mails gepflegt, sondern im Rahmen des Dialogs strukturiert erfasst und verarbeitet werden. Wiederkehrende Tätigkeiten wie das Öffnen von Vorlagen, das manuelle Einfügen von Daten und das Speichern von Dateien werden weitgehend in automatisierte Abläufe verlagert, auch wenn die dialogbasierte Erfassung insbesondere bei der Neuanlage von Kunden noch relativ zeitintensiv ist.

## Forschungsfrage 3

*Wie kann eine ergänzende Web-Anwendung gestaltet werden, die einen strukturierten Zugriff auf Kunden, Rechnungen und Dokumente ermöglicht und die im Chat erfassten Daten für die weitere Verwaltung nutzbar macht?*

Die entwickelte Web-Anwendung zeigt, dass eine schlanke, auf Anzeige- und Filterfunktionen fokussierte Oberfläche ausreicht, um die im Chat erfassten Daten für die Verwaltung nutzbar zu machen. Durch die Darstellung von Kunden, Rechnungen und Dokumenten auf Basis einer gemeinsamen, konsistenten Datenstruktur wird ein schneller Überblick über den aktuellen Stand ermöglicht. Nutzende können erzeugte Rechnungen einsehen, heruntergeladene Dokumente auffinden und Zusammenhänge zwischen Kundendaten und zugehörigen Belegen nachvollziehen, ohne direkt mit den zugrunde liegenden Cloud-Diensten interagieren zu müssen.

### Forschungsfrage 4

*Inwieweit zeigen sich beim praktischen Einsatz des prototypisch umgesetzten Systems Grenzen, Herausforderungen und Verbesserungspotenziale in Bezug auf Nutzbarkeit, Prozessdurchlaufzeiten und technische Robustheit?*

Die Evaluation macht deutlich, dass die Kernprozesse zur Rechnungserstellung und Dokumentenablage funktional stabil ablaufen, die wahrgenommenen Prozessdurchlaufzeiten aus Sicht Einzelner jedoch teilweise als lang empfunden werden. Insbesondere die Vielzahl einzelner Dialogschritte bei der Datenerfassung und die sequentielle Abarbeitung der Automatisierungsworkflows tragen zu spürbaren Wartezeiten bei. Hinsichtlich der technischen Robustheit zeigt sich, dass typische Eingabefehler abgefangen und Dialoge in vielen Fällen sinnvoll fortgeführt werden, komplexere Sonderfälle jedoch noch nicht vollständig abgedeckt sind. Daraus ergeben sich konkrete Verbesserungspotenziale, etwa in der Straffung der Dialogführung, der Erweiterung der Validierungslogik und der Optimierung der zugrunde liegenden Integrationsprozesse.

### 7.3 Beitrag der Arbeit und praktische Implikationen

Die vorliegende Arbeit leistet einen Beitrag zur praxisnahen Digitalisierung administrativer Prozesse in kleinen Dienstleistungsunternehmen und bei Selbstständigen. Im Unterschied zu klassischen Ansätzen, die auf umfassende Buchhaltungs- oder ERPs-Systeme setzen, verfolgt sie einen niedrigschwlligen, prototypischen Lösungsweg, der auf allgemein verfügbare Cloud-Dienste und einen bereits etablierten Kommunikationskanal aufbaut. Durch die Kombination eines Chatbots mit einer Integrationsplattform, einer cloudbasierten Datenhaltung und einer schlanken Web-Anwendung wird gezeigt, wie sich mit begrenztem technischen Aufwand ein durchgängiger Prozess zur Erstellung und Ablage von Rechnungen und Dokumenten realisieren lässt.

Aus wissenschaftlicher Perspektive besteht der Beitrag vor allem in der Konzeption und prototypischen Umsetzung eines integrierten Systems, das dialogbasierte Interaktion mit automatisierter Dokumentenerzeugung und strukturierter Ablage verbindet. Die Arbeit verdeutlicht, welche architektonischen Entscheidungen und Integrationsmuster sich für diesen Anwendungsfall eignen und wie bestehende Dienste so kombiniert werden können, dass Medienbrüche reduziert und wiederkehrende manuelle Tätigkeiten teilweise automatisiert werden. Darüber hinaus liefert die Evaluation Einblicke in die Grenzen eines

solchen Ansatzes, etwa im Hinblick auf Prozessdurchlaufzeiten, Fehlerbehandlung und Abhängigkeiten von Drittanbietern, und schafft damit eine Grundlage für weiterführende Untersuchungen.

Für die Praxis ergeben sich mehrere Implikationen. Zum einen zeigt der Prototyp, dass kleine Unternehmen und Selbstständige ihre Rechnungs- und Dokumentenprozesse verbessern können, ohne sofort in komplexe oder kostspielige Spezialsoftware investieren zu müssen. Stattdessen können sie auf bereits genutzte Kommunikationskanäle und Cloud-Dienste zurückgreifen und diese gezielt zu einem sinnvoll orchestrierten Prozess verbinden. Zum anderen macht die Arbeit deutlich, dass insbesondere die Gestaltung der Dialoge und der Benutzerführung entscheidend dafür ist, ob ein solches System als Unterstützung und nicht als zusätzliche Hürde wahrgenommen wird. Die identifizierten Verbesserungspotenziale – etwa eine kompaktere Dialogführung, erweiterte Validierungen oder eine engere Verzahnung mit bestehenden buchhalterischen Abläufen – geben konkrete Hinweise darauf, wie vergleichbare Lösungen in der Praxis weiterentwickelt und schrittweise professionalisiert werden können.

## 7.4 Ausblick und Weiterentwicklung

Die im Rahmen dieser Arbeit entwickelte Lösung zeigt, dass sich mit standardisierten Cloud-Diensten und einem Chatbot-basierten Ansatz ein durchgängiger Prozess zur Erstellung und Ablage von Rechnungen und administrativen Dokumenten realisieren lässt. Gleichzeitig wird deutlich, dass der Prototyp in mehreren Bereichen gezielt weiterentwickelt werden kann, um sowohl die Nutzbarkeit als auch die Automatisierungstiefe zu erhöhen.

Ein zentrales Entwicklungsfeld betrifft die Optimierung der Dialogführung und der Prozessdurchlaufzeiten. Die schrittweise Erfassung zahlreicher Datenpunkte, insbesondere bei der Neuanlage von Kunden, führt zu vergleichsweise langen und potenziell ermüdenden Interaktionen. Künftige Arbeiten könnten hier ansetzen, indem Eingaben stärker gebündelt, geeignete Voreinstellungen genutzt oder bestehende Informationen aus früheren Interaktionen wiederverwendet werden. Ergänzend bietet sich eine weitergehende Validierung von Eingaben sowie eine verbesserte Fehlertoleranz an, um Sonderfälle robuster abzufangen und Unterbrechungen im Ablauf zu vermeiden.

Darüber hinaus eröffnet die Integration mit weiteren Systemen und Datenquellen zusätzliche Potenziale. Denkbar sind etwa Schnittstellen zu Buchhaltungs- oder Steuersoftware,

um erzeugte Rechnungen automatisiert zu übergeben, oder der Zugriff auf Stammdaten aus bestehenden Kundensystemen, um Doppeleingaben zu vermeiden. Auch eine Erweiterung der Lösung auf weitere Dokumenttypen und administrative Prozesse – beispielsweise Angebote, Mahnungen oder vertragsbezogene Unterlagen – könnte den Nutzen des Ansatzes für kleine Unternehmen erhöhen und zu einem umfassenderen, modular erweiterbaren Administrationswerkzeug führen.

Schließlich stellen Skalierbarkeit, Betriebssicherheit und langfristige Wartbarkeit wichtige Themen für eine produktive Nutzung dar. Zukünftige Arbeiten könnten untersuchen, wie sich Abhängigkeiten von einzelnen Cloud-Diensten reduzieren, Monitoring- und Logging-Konzepte ausbauen und Sicherheits- sowie Datenschutzanforderungen systematisch adressieren lassen. Auf dieser Grundlage wäre es möglich, den in dieser Arbeit entwickelten Prototyp schrittweise von einer experimentellen Lösung zu einem belastbaren System weiterzuentwickeln, das in der Praxis dauerhaft eingesetzt werden kann.

# Literaturverzeichnis

- [1] : *ISO 9241-11: Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts.* 2018. – URL <https://www.iso.org/standard/63500.html>. – Zugriffsdatum: 2026-02-01
- [2] AIRTABLE: *Formula Field Overview.* 2025. – URL <https://support.airtable.com/docs/formula-field-overview>. – Zugriffsdatum: 2025-12-26. – Offizielle Dokumentation zu Feldern, Formeln und Datenstrukturen in Airtable, inkl. Nutzung in Automatisierungen
- [3] AIRTABLE: *Getting started with Airtable's Web API.* 2025. – URL <https://support.airtable.com/docs/getting-started-with-airtables-web-api>. – Zugriffsdatum: 2025-12-27. – Dokumentation zur Web-API von Airtable, inklusive Erstellen, Lesen, Aktualisieren und Löschen von Datensätzen
- [4] AIRTABLE: *Formula Field Reference.* 2026. – URL <https://support.airtable.com/docs/formula-field-reference>. – Zugriffsdatum: 2026-02-01. – Dokumentation der Airtable-Formeln wie FIND(), LOWER(), IF()
- [5] BALZERT, Helmut: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb.* Spektrum, 2011
- [6] BAVARESCO, Renan ; SILVEIRA, Diego ; REIS, Jorge ; PRATES, Ruy u. a.: Service Chatbots: A Systematic Review. In: *Expert Systems with Applications* 165 (2021), S. 113806. – URL <https://www.sciencedirect.com/science/article/pii/S0957417421008745>. – Systematische Übersicht zu Service-Chatbots, inkl. Definition, Typen, technischen Ansätzen und typischen Architekturen
- [7] BECK IT GMBH: *Die manuelle und automatisierte Rechnungsverarbeitung im Vergleich.* 2022. – URL <https://beck-it.com/news/die-manuelle-und-automatisierte-rechnungsverarbeitung-im-vergleich/>. – Zugriffsdatum: 2025-12-27

- [8] BITKOM E.V.: *Digitalisierung im Mittelstand – Status quo und Herausforderungen*. 2023. – URL <https://www.bitkom.org/Presse/Presseinformation/Digitalisierung-im-Mittelstand>. – Zugriffssdatum: 2026-01-21
- [9] BITKOM E.V. ; FORUM ELEKTRONISCHE RECHNUNG DEUTSCHLAND (FERD): *Elektronische Beleg- und Rechnungsdaten im Jahr 2022*. 2022. – URL [https://www.bitkom.org/sites/main/files/2022-08/220808\\_Elektronische%20Beleg-%20und%20Rechnungsdaten\\_final.pdf](https://www.bitkom.org/sites/main/files/2022-08/220808_Elektronische%20Beleg-%20und%20Rechnungsdaten_final.pdf). – Zugriffssdatum: 2025-12-27
- [10] BUSINESS SOLUTIONS GMBH: *Digitale Rechnungsverarbeitung: Ein Überblick*. 2021. – URL <https://www.business-solutions.gmbh/digitalisierung-wissen/digitale-rechnungsverarbeitung>. – Zugriffssdatum: 2026-01-14. – Überblicksartikel zur digitalen Rechnungsverarbeitung, der Vorteile wie automatisierte Datenerfassung, Workflows, digitale Archivierung und geringere Fehleranfälligkeit beschreibt
- [11] EBNER STOLZ: *Künstliche Intelligenz im Rechnungswesen: Potenziale und Grenzen*. 2023. – URL [https://www.ebnerstolz.de/pdfs/03/6/1/6/0/5/STUDIE\\_KI\\_RECHNUNGSWESEN\\_JAHRESABSCHLUSS.pdf](https://www.ebnerstolz.de/pdfs/03/6/1/6/0/5/STUDIE_KI_RECHNUNGSWESEN_JAHRESABSCHLUSS.pdf). – Zugriffssdatum: 2026-01-14. – Studie zu Einsatz und Effekten von KI-Lösungen im Rechnungswesen, inkl. Effizienzgewinnen und Fehlerreduktion
- [12] EUROPÄISCHE UNION: *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates*. 2016. – URL <https://eur-lex.europa.eu/eli/reg/2016/679/oj>. – Zugriffssdatum: 2025-01-15. – Datenschutz-Grundverordnung (DSGVO)
- [13] GOOGLE: *Exporting Google Docs as PDF using the Google Drive API*. 2024. – URL <https://developers.google.com/drive/api/guides/ref-export-formats>. – Zugriffssdatum: 2025-01-30
- [14] GOOGLE: *Merge Data into Google Docs*. 2024. – URL <https://developers.google.com/docs/api/how-tos/merge>. – Zugriffssdatum: 2026-02-01. – Offizielle Dokumentation zum Platzhalter-basierten Zusammenführen von Daten in Google Docs (Mail-Merge-Prinzip)
- [15] GOOGLE: *Organize your files in Google Drive*. 2024. – URL <https://support.google.com/drive/answer/2424384>. – Zugriffssdatum: 2026-02-01. – Best Practices zur Ordnerstruktur und Dateiorganisation in Google Drive

- [16] GOOGLE: *Google Docs API overview*. 2025. – URL <https://developers.google.com/workspace/docs/api/how-tos/overview>. – Zugriffsdatum: 2025-12-27. – Überblick über Funktionen zur programmgesteuerten Erstellung und Bearbeitung von Google-Dokumenten, inkl. Nutzung von Vorlagen und Platzhaltern
- [17] GOOGLE: *Google Drive API overview*. 2025. – URL <https://developers.google.com/workspace/drive/api/guides/about-sdk>. – Zugriffsdatum: 2025-12-27. – Überblick über Funktionen zum Anlegen, Organisieren und Abrufen von Dateien und Ordnern in Google Drive per API
- [18] GOOGLE LLC: *Method: files.export — Google Drive API*. 2025. – URL <https://developers.google.com/workspace/drive/api/reference/rest/v3/files/export>. – Zugriffsdatum: 2025-12-27. – Beschreibung der Methode `files.export` der Google Drive API, mit der Google-Docs-Dokumente unter anderem als PDF exportiert werden können
- [19] INSTITUT FÜR MITTELSTANDSFORSCHUNG BONN: *Digitalisierung von Geschäftsprozessen im Mittelstand*. 2022. – URL <https://www.ifm-bonn.org/statistiken/digitalisierung>. – Zugriffsdatum: 2026-01-21
- [20] ISO/IEC: *ISO/IEC 25010:2023 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model*. INTERNATIONAL STANDARD. 2023. – URL <https://www.iso.org/standard/78176.html>. – Zugriffsdatum: 2025-12-27. – Product quality model mit Qualitätsmerkmalen wie Functional Suitability, Performance Efficiency, Compatibility, Reliability, Security, Maintainability, etc.
- [21] JOHANN, Peter: *Nicht-funktionale Anforderungen*. 2025. – URL <https://www.peterjohann-consulting.de/nicht-funktionale-anforderungen/>. – Zugriffsdatum: 2025-12-13
- [22] MAKE: *Airtable - Apps Documentation*. 2025. – URL <https://apps.make.com/airtable>. – Zugriffsdatum: 2025-12-27. – Dokumentation der Airtable-App in Make, inklusive Einrichtung der Verbindung und Verwendung der Module zur Verwaltung von Datensätzen
- [23] MAKE: *Flow control (Aggregator modules)*. 2025. – URL <https://help.make.com/flow-control>. – Zugriffsdatum: 2026-01-20

- [24] MAKE.COM: *Google Docs Integration*. 2024. – URL <https://www.make.com/en/integrations/google-docs>. – Zugriffssdatum: 2025-01-30
- [25] MAKE.COM: *WhatsApp Business Cloud — Modules Documentation*. 2025. – URL <https://apps.make.com/whatsapp-business-cloud-modules>. – Zugriffssdatum: 2026-01-16. – Dokumentation der Make-Plattform am Beispiel der WhatsApp-Business-Cloud-Module, inkl. Triggern, Aktionen und Szenariokonzept
- [26] MAKE.COM: *WhatsApp Cloud API - Send Document Messages*. <https://www.make.com/en/help/app/whatsapp-cloud-api>. 2025. – Make.com WhatsApp Integration (Modul: Send a message)
- [27] MAKE.COM: *Airtable modules (Make.com) — Search records*. 2026. – URL <https://apps.make.com/airtable-modules>. – Zugriffssdatum: 2026-02-01. – Abschnitt "Search records"
- [28] MAKE.COM: *Google Docs modules (Make.com)*. 2026. – URL <https://apps.make.com/google-docs>. – Zugriffssdatum: 2026-02-01. – Dokumenterstellung und Template-Verarbeitung
- [29] MAKE.COM: *Google Drive modules (Make.com)*. 2026. – URL <https://apps.make.com/google-drive>. – Zugriffssdatum: 2026-02-01. – Dokumentenablage, Ordnererstellung und Datei-Upload
- [30] MELL, Peter ; GRANCE, Timothy: The NIST Definition of Cloud Computing / National Institute of Standards and Technology. NIST, 2011 (Special Publication 800-145). – Forschungsbericht. – URL <https://csrc.nist.gov/pubs/sp/800/145/final>. – Zugriffssdatum: 2026-01-14. Standarddefinition von Cloud Computing mit zentralen Merkmalen wie On-Demand-Self-Service, Broad Network Access und Resource Pooling
- [31] META PLATFORMS, INC.: *Media — WhatsApp Business Platform Cloud API*. 2025. – URL <https://developers.facebook.com/docs/whatsapp/cloud-api/>. – Zugriffssdatum: 2025-12-27. – Dokumentation der WhatsApp Business Platform Cloud API, inklusive Senden und Empfangen von Nachrichten sowie Upload und Abruf von Mediendateien
- [32] NEWMAN, Michael S.: *Single Page Web Applications: JavaScript End to End*. Manning Publications, 2014. – URL <https://www.manning.com/books/single-page-web-applications>. – Praxisorientiertes Buch zu Architektur, Design

und Implementierung moderner Single-Page-Webanwendungen mit JavaScript, inkl. Client–Server-Modell, REST-APIs und Sicherheitsaspekten

- [33] PLEO TECHNOLOGIES APS: *Automatisierte Rechnungsverarbeitung: Warum sie durch die E-Rechnung immer wichtiger wird.* 2024. – URL <https://blog.pleo.io/de/automatisierte-rechnungsverarbeitung/>. – Zugriffsdatum: 2025-12-27
- [34] SEIBERT MEDIA GMBH: *Qualität, funktionale und nichtfunktionale Anforderungen in der Software-Entwicklung.* 2018. – URL <https://seibert.group/blog/2018/05/14/qualitaet-funktionale-und-nichtfunktionale-anforderungen-in-der-software-entwicklung/>. – Zugriffsdatum: 2025-12-27
- [35] 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
- [36] SUPABASE: *Edge Functions.* 2026. – URL <https://supabase.com/docs/guides/functions>. – Zugriffsdatum: 2026-02-01. – Dokumentation zu serverseitigen Edge Functions auf Basis von Deno
- [37] VOIGT, Paul ; BUSSCHE, Axel von dem: The EU General Data Protection Regulation (GDPR): A Practical Guide. In: *Springer International Publishing* (2017)
- [38] ZENWORK: *The Benefits of Automated Invoicing for Small Business Growth.* 2025. – URL <https://www.zenwork.com/payments/blog/benefits-of-invoicing-automation/>. – Zugriffsdatum: 2026-01-20

# 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 <small>A</small> T <small>E</small> X	Textsatz- und Layout-Werkzeug zur Erstellung dieser Bachelorarbeit
Make.com	Implementierung der Chatbot-Workflows und Automatisierungslogik
Airtable	Zentrale cloudbasierte Datenhaltung für Kunden- und Rechnungsdaten
Google Docs	Vorlage und automatisierte Generierung von Rechnungsdocumenten
Google Drive	Strukturierte Ablage der erzeugten Rechnungen und Dokumente
Lovable Cloud / Supabase Edge Functions	Hosting der Web-Anwendung und serverlose Ausführung von Integrationslogik
React	Umsetzung der ClientHub-Webanwendung als Single-Page-Application
TypeScript	Typisierte Implementierung der Frontend-Logik
Tailwind CSS	Styling und Layout der Webanwendung
shadcn/ui	UI-Komponentenbibliothek für wiederverwendbare Interface-Elemente
Visual Studio Code	Entwicklungsumgebung für Frontend- und Edge-Function-Code
ChatGPT	Unterstützung bei Konzeption, Textstrukturierung und sprachlicher Überarbeitung
Perplexity AI	Rechercheunterstützung und Zusammenfassung technischer Sachverhalte



## **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