HAW HAMBURG

BACHELOR THESIS
Framarz Alizadeh

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

FAKULTÄT INFORMATIK UND DIGITALE GESELLSCHAFT

Faculty of Computer Science and Digital Society

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Framarz Alizadeh

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

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

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

Eingereicht am: 07. Juni 1954

Framarz Alizadeh

Thema der Arbeit

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

Stichwörter

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

Kurzzusammenfassung

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

Framarz Alizadeh

Title of Thesis

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

Keywords

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

Abstract

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

Inhaltsverzeichnis

Abbildungsverzeichnis **xi**

Tabellenverzeichnis **xiii**

1 Einleitung **1**
1.1 Problemstellung 1
1.2 Motivation und Zielsetzung 1
1.3 Forschungsfragen 1
1.4 Vorgehensweise und Aufbau der Arbeit 1

2 Theoretische Grundlagen **2**
2.1 Digitalisierung administrativer Prozesse und Rechnungswesen 2
2.2 Chatbots und dialogbasierte Systeme 4
2.3 Cloudbasierte Datenhaltung und Integrationen 6
2.4 Dokumentengenerierung in der Cloud 7
2.5 Grundlagen moderner Webanwendungen 8

3 Technologischer Hintergrund **9**
3.1 Make.com als Integrations- und Automatisierungsplattform 9
3.2 Airtable als Datenhaltung 9
3.3 Google Docs und Google Drive 9
3.4 Lovable Cloud / Supabase Edge Functions 9
3.5 Frontend-Stack (React, TypeScript, Tailwind, shadcn/ui) 9

4 Konzeption des Systems **10**
4.1 Anforderungen und Use Cases 10
4.2 Zielarchitektur und Systemübersicht 13
4.3 Datenmodell und Datenflüsse 15
4.4 Prozessablauf vom Chatbot bis zur Ablage 18
4.4.1 Ablauf „Rechnung neu erstellen“ 18

4.4.2 Ablauf „Dokument ablegen“ 21
4.5 Sicherheits- und Datenschutzkonzept 22

5 Implementierung **23**
5.1 Implementierung des Chatbots (make.com-Szenarien) 23
5.2 Datenhaltung und Web-Anwendung (Airtable und ClientHub) 26
5.2.1 Tabellenstruktur und Datenmodell in Airtable 26
5.2.2 Interaktion zwischen Chatbot und Datenhaltung 27
5.2.3 Architektur der Web-Anwendung (ClientHub) 28
5.2.4 Funktionale Abgrenzung zwischen Web-App und Chatbot 29
5.3 Automatische Dokumentenerstellung in Google Docs 29
5.3.1 Rechnungsvorlage und Platzhalter 30
5.3.2 Generierung und Versand der PDF-Rechnung 30
5.4 Ablagestrategie in Google Drive (Jahr/Quartal/Monat) 31
5.5 Implementierung der ClientHub WebApp 32
5.5.1 Frontend (React, UI-Konzept) 32
5.5.2 Edge Functions in Lovable Cloud 32
5.5.3 API-Anbindung an Airtable und Google Drive 32

6 Evaluation **33**
6.1 Testkonzept und Testumgebung 33
6.2 Funktionale Tests 33
6.3 Performance-Analyse 33
6.4 Vergleich: manueller vs. automatisierter Prozess 33
6.5 Usability-Bewertung 33
6.6 Grenzen des Systems und Risiken 33

7 Fazit und Ausblick **34**
7.1 Zusammenfassung der Ergebnisse 34
7.2 Beantwortung der Forschungsfragen 34
7.3 Beitrag der Arbeit und praktische Implikationen 34
7.4 Ausblick und Weiterentwicklung 34

Literaturverzeichnis **35**

A Anhang **38**
A.1 Verwendete Hilfsmittel 38

Selbstständigkeitserklärung **39**

Abbildungsverzeichnis

Tabellenverzeichnis

A.1 Verwendete Hilfsmittel und Werkzeuge 38

1 Einleitung

1.1 Problemstellung

1.2 Motivation und Zielsetzung

1.3 Forschungsfragen

1.4 Vorgehensweise und Aufbau der Arbeit

2 Theoretische Grundlagen

2.1 Digitalisierung administrativer Prozesse und Rechnungswesen

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 und wenig strukturierte Ablagesysteme. 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. Sie ermöglicht es, den manuellen Aufwand im agesgeschäft zu reduzieren und gleichzeitig die Qualität sowie 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.

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. Rechnungen und zugehörige Dokumente werden zentral abgelegt und eindeutig Kunden sowie Zeiträumen zugeordnet. 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 und 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.

In Bezug auf das Szenario sind vor allem die Informationen entscheidend, die nötig 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 System, das in dieser Arbeit entworfen wurde, 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 und eine revisionssichere Ablage der erzeugten Belege. Es werden weder Hauptbücher geführt noch Buchungssätze erstellt. Aufgaben wie die Kontierung, die Umsatzsteuervoranmeldung oder der Jahresabschluss bleiben nach wie vor bei Steuerberatungen und spezialisierten Buchhaltungssystemen. Sie 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 zur Automatisierung von Finanzprozessen mit Chatbots zeigen, dass dialogbasierte Systeme insbesondere bei der Verarbeitung von Rechnungs- und Ausgabendaten signifikante Effizienzgewinne erzielen und gleichzeitig die Fehlerquote verringern können.

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 statis-

tische Sprachmodelle, um Kontext zu berücksichtigen und flexibel auf unterschiedliche Eingaben reagieren zu können.

Früher Chatbots waren 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.

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 genutzt werden. 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.

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 werden sie 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.

2.3 Cloudbasierte Datenhaltung und Integrationen

Cloudbasierte Datenhaltung bezeichnet die Speicherung von Informationen in extern betriebenen Rechenzentren, auf die über das Internet zugegriffen wird. 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 Automatisierungsworkflows 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 Bürosoftware 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. 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 hin zu nicht veränderbaren Formaten wie PDF, die sich für den Versand an Kunden und 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 Formatierungsarbeit erforderlich sind.

2.5 Grundlagen moderner Webanwendungen

Moderne Webanwendungen unterscheiden sich deutlich von klassischen, statischen Websites. Während früher hauptsächlich einzelne HTML-Seiten ausgeliefert wurden, handeln aktuelle Anwendungen komplexe Geschäftslogik, dynamische Daten und interaktive Benutzeroberflächen im Browser. Grundlage ist dabei das Client–Server-Modell: Ein Webbrowser fungiert als Client, der über das 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 (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 Sortierfunktionen, komfortabel abbilden.

Die serverseitige Ebene moderner Webanwendungen stellt meist klar definierte Programmierschnittstellen bereit, häufig in Form von 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.

3 Technologischer Hintergrund

3.1 Make.com als Integrations- und Automatisierungsplattform

3.2 Airtable als Datenhaltung

3.3 Google Docs und Google Drive

3.4 Lovable Cloud / Supabase Edge Functions

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

4 Konzeption des Systems

4.1 Anforderungen und Use Cases

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

Das geplante System richtet sich vor allem an kleine Dienstleistungsfirmen, Einzelunternehmer sowie kleine und mittlere Gewerbebetriebe. Diese müssen regelmäßig Rechnungen erstellen und verwalten, verfügen jedoch weder über die Ressourcen noch über den Bedarf für umfassende Buchhaltungs- oder ERP-Systeme. Häufig mangelt es diesen Personen an IT-Kenntnissen und sie haben wenig Zeit. Deshalb benötigen sie einfache, zuverlässige und größtenteils automatisierte Prozesse. Auch Steuerberaterinnen und Steuerberater sind auf vollständig und strukturiert archivierte Rechnungsunterlagen angewiesen. Die Erkenntnisse aus Studien und Praxisberichten zeigen, dass in kleinen Unternehmen manuelle Rechnungsprozesse besonders zeitaufwendig und fehleranfällig sind. Außerdem resultieren sie häufig in Medienbrüchen und einem Mangel an Transparenz [4, 22, 9].

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

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

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

Ziele und Anforderungen aus Sicht der Anwender

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

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

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

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

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

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

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

Da personenbezogene Daten und Abrechnungsinformationen verarbeitet werden, kommt der Datensicherheit eine besondere Bedeutung zu. Zugangsdaten sind sicher zu speichern, die Kommunikation ist zu verschlüsseln und der Zugriff auf autorisierte Nutzer zu beschränken. Weiterhin soll die Systemarchitektur modular und erweiterbar gestaltet sein, um zukünftige Anforderungen oder zusätzliche Funktionen mit vertretbarem Aufwand integrieren zu können. Insgesamt orientiert sich das System damit an anerkannten Qualitätsmerkmalen nicht-funktionaler Anforderungen, wie sie in der Literatur beschrieben sind [21, 14].

Anwendungsfall „Rechnung neu erstellen“

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

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

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

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

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

4.2 Zielarchitektur und Systemübersicht

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

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

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

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

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

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

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

Wartungsaufwand benötigen und dennoch eine Lösung suchen, die flexibel mit wachsenden Anforderungen erweitert werden kann. Lose Kopplung, Modularität und eine klare Schichtentrennung gelten in der Softwarearchitektur als wesentliche Voraussetzungen für wartbare und erweiterbare Systeme. [3, 7, 8]. Der konsequente Einsatz wohldefinierter APIs wird als zentraler Baustein integrationsfähiger, verteilter Systeme beschrieben [23].

4.3 Datenmodell und Datenflüsse

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

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

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

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

Ein vergleichbares Vorgehen wird bei der Neuanlage von Kunden angewendet. Temporäre Felder wie Vorname, Nachname, Adresse, Postleitzahl und Umsatzsteuer-Identifikationsnummer werden zunächst schrittweise erfasst und können bei Bedarf angepasst werden. Erst nachdem alle erforderlichen Felder vollständig ausgefüllt und eine automatisch generierte Zusammenfassung der Kundendaten explizit bestätigt wurde, erfolgt die dauerhafte Speicherung des Kunden in einer separaten Kundentabelle. Auch rechnungsbezogene Parameter wie Leistungsbeschreibung, Menge und Einzelpreis werden zunächst in „User_Sessions“ zwischengespeichert. „User_Sessions“ fungiert somit als transiente 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 „Customer“ verwaltet, die als zentrale Kundendatenbank des Systems dient. Neue Kundendatensätze können sowohl über die Webanwendung als auch direkt über den Chatbot angelegt werden. In dieser Tabelle werden alle für die Rechnungsstellung relevanten Stammdaten gespeichert, darunter Vor- und Nachname, Firmenbezeichnung, vollständige Adresse, E-Mail-Adresse, Telefonnummer sowie die Umsatzsteuer-Identifikationsnummer. Nach Abschluss eines Neuanlage-Dialogs werden die zuvor in „User_Sessions“ erfassten temporären Kundendaten in einen konsistenten Datensatz in „Customer“ überführt. Bestehende Kunden werden über ihre in der Tabelle hinterlegten Attribute identifiziert und können im Dialog ausgewählt werden, sodass die bereits gespeicherten Stammdaten ohne erneute Eingabe für die Rechnungserstellung genutzt werden.

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

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

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

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

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

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

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

4.4 Prozessablauf vom Chatbot bis zur Ablage

4.4.1 Ablauf „Rechnung neu erstellen“

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

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

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

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

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

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

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

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

4.4.2 Ablauf „Dokument ablegen“

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

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

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

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

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

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

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

4.5 Sicherheits- und Datenschutzkonzept

5 Implementierung

5.1 Implementierung des Chatbots (make.com-Szenarien)

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, 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 [19]. Jedes empfangene Ereignis initiiert einen eindeutigen Ablauf, wodurch eine synchrone Verarbeitung der Nachrichten sichergestellt wird.

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 [18].

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 [19].

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 [6].

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 [18].

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 [6].

- **Kein Treffer:** Der Nutzer kann abbrechen, neu suchen oder einen neuen Kunden erstellen.
- **Ein Treffer:** Der Kunde kann bestätigt oder eine erneute Suche ausgelöst werden.

- **Mehrere Treffer:** Der Nutzer wählt per nummeriertem Emoji den gewünschten Datensatz aus.

Nach Auswahl oder Neuanlage eines Kunden werden die zugehörigen Stammdaten in `Customer` gespeichert und zugleich als Relation in `Rechnung` übernommen. Der folgende Dialog sammelt schrittweise rechnungsbezogene Informationen (Datum, Rechnungsnummer, Leistungsbeschreibung, Menge, Einzelpreis, Zahlungsfrist, Zahlart). Die Dialogschritte sind durch `current_step`-Werte (z. B. *Datum ausgewählt*, *Position erstellt*) klar strukturiert. Eingaben werden jeweils kurz bestätigt, bevor das Szenario in den nächsten Schritt wechselt.

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

Nach der Positionsbestätigung fragt der Bot nach Menge und Einzelpreis. Beide Werte werden verifiziert, in temporären Feldern von `User_Sessions` zwischengespeichert und anschließend in den Tabellen `Rechnung` und `Rechnungsposition` persistiert. Abschließend erstellt das Szenario mit der Google-Docs-Integration ein Rechnungsdokument auf Basis eines Templates, fügt alle dynamischen Felder ein, exportiert das Dokument als PDF und sendet es dem Nutzer direkt über WhatsApp zurück [16]. 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 [20].

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 [**?**]. 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 [1].

5.2.1 Tabellenstruktur und Datenmodell in Airtable

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

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 [**?**].

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 [2].

Die Tabelle `Rechnung` speichert kopfbezogene Rechnungsdaten wie Kundendaten, Rechnungsdatum, Rechnungsnummer, Fälligkeit, Zahlvariante sowie Gesamt- und Steuerbeträge. Die zugehörigen Einzelpositionen werden in der Tabelle `Rechnungsposition` 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 [**?** 18]. 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`, `Rechnung` und `Rechnungsposition` 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 [1]. 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 [24].

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 [24, 15]. 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 [24]. 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 Tabellenform, 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 [11]. 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`, `Rechnung` und `Rechnungsposition` zugreifen, bleiben Stamm- und Belegdaten konsistent, unabhängig davon, ob sie über den WhatsApp-Dialog oder die Web-Anwendung angelegt oder verändert werden [1]. 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 [17]. 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 [10].

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 [12]. 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 `Rechnung` 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 Gesamtbeträge (`{{company_subtotal}}`, `{{company_total}}`) zum Einsatz, sodass sowohl Einzelpositionen als auch aggregierte Werte dynamisch befüllt werden können [13].

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 [12].

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`, `Rechnung` und `Rechnungsposition` ausgelesen und so aufbereitet, dass für jeden im Dokument definierten Platzhalter ein eindeutiger Wert vorliegt [**?**]. 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 [17].

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 [10, 5]. 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 [**?** 11].

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 verwalten zu können [**?**]. Alle vom Nutzer hochgeladenen Dokumente werden im Pfad „Dokument speichern“ des Chatbots systematisch nach Jahr, Quartal und Monat einsortiert.

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-Uploads 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 in Google Drive schrittweise, ob die benötigte Ordnerhierarchie bereits existiert, und legt fehlende Ebenen automatisch an. Zunächst wird ein kundenbezogener Stammordner identifiziert, der in der Regel die Telefonnummer des Nutzers als Ordnungsmerkmal verwendet; existiert dieser Ordner nicht, wird er erstellt und als Wurzel für alle weiteren Unterordner verwendet [**?**]. Anschließend wird innerhalb dieses Stammordners geprüft, ob ein Unterordner für das gewählte Jahr vorhanden ist; falls nicht, wird der entsprechende Jahresordner angelegt, bevor die Verarbeitung mit den nächsten Ebenen fortgesetzt wird. Nach demselben Muster werden nacheinander die Unterordner für Quartal und Monat geprüft und bei Bedarf erstellt [**?**].

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 [**?**].

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 [5]. 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)

5.5.2 Edge Functions in Lovable Cloud

5.5.3 API-Anbindung an Airtable und Google Drive

6 Evaluation

6.1 Testkonzept und Testumgebung

6.2 Funktionale Tests

6.3 Performance-Analyse

6.4 Vergleich: manueller vs. automatisierter Prozess

6.5 Usability-Bewertung

6.6 Grenzen des Systems und Risiken

7 Fazit und Ausblick

7.1 Zusammenfassung der Ergebnisse

7.2 Beantwortung der Forschungsfragen

7.3 Beitrag der Arbeit und praktische Implikationen

7.4 Ausblick und Weiterentwicklung

Literaturverzeichnis

[1] Airtable: *Formula Field Overview*. 2025. – URL https://support.airtable.com/docs/formula-field-overview. – Zugriffsdatum: 2025-12-26. – Überblick über Formelfelder und API-Nutzung in Airtable

[2] Airtable: *Formula Field Reference*. 2025. – URL https://support.airtable.com/docs/formula-field-reference. – Zugriffsdatum: 2025-12-26. – Referenz zu Funktionen wie LOWER und FIND für Textsuche

[3] Balzert, Helmut: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum, 2011

[4] Beck-IT: *Die manuelle und automatisierte Rechnungsverarbeitung im Vergleich*. 2022. – URL https://beck-it.com/news/die-manuelle-und-automatisierte-rechnungsverarbeitung-im-vergleich/. – Zugriffsdatum: 2025-12-13

[5] Consultevo: *Automate Google Docs to PDF with Make.com*. 2025. – URL https://consultevo.com/make-com-google-doc-to-pdf-automation/. – Zugriffsdatum: 2025-12-27. – Praxisbeispiel für die automatische Konvertierung von Google Docs in PDF

[6] ConsultEvo: *Mastering Make.com Array Aggregator*. 2025. – URL https://consultevo.com/make-com-array-aggregator-guide/. – Zugriffsdatum: 2025-12-26

[7] dpunkt.verlag: *Grundlagen des modularen Softwareentwurfs*. 2020. – URL https://www.assets.dpunkt.de/openbooks/Openbook_Spichale_API-Design_2A.pdf. – Zugriffsdatum: 2025-12-13

[8] dpunkt.verlag: *Langlebige Softwarearchitekturen – Erweiterungen*. 2020. – URL https://www.langlebige-softwarearchitekturen.de/media/wps/

uploads/langlebige-softwarearchitekturen-erweiterungen.pdf. – Zugriffsdatum: 2025-12-13

[9] Flowwer: *Automatisierte Rechnungsverarbeitung: Effizient & fehlerfrei.* 2025. – URL https://www.flowwer.de/automatisierte-rechnungsverarbeitung-der-beste-weg-zum-erfolg/. – Zugriffsdatum: 2025-12-13

[10] 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 Bearbeitung von Google-Dokumenten

[11] Google: *Google Drive API overview.* 2025. – URL https://developers.google.com/workspace/drive/api/guides/about-sdk. – Zugriffsdatum: 2025-12-26. – Überblick über Funktionen zum Suchen, Hoch- und Herunterladen von Dateien in Google Drive

[12] Google: *Mail merge with Docs API.* 2025. – URL https://developers.google.com/workspace/docs/api/samples/mail-merge. – Zugriffsdatum: 2025-12-27. – Beispiel für die Erzeugung personalisierter Dokumente aus einer Vorlage

[13] Google: *Merge text into a document.* 2025. – URL https://developers.google.com/workspace/docs/api/how-tos/merge. – Zugriffsdatum: 2025-12-27. – Beschreibung des Ersetzens von Platzhaltern (Mail-Merge) in Google Docs

[14] Johann, Peter: *Nicht-funktionale Anforderungen.* 2025. – URL https://www.peterjohann-consulting.de/nicht-funktionale-anforderungen/. – Zugriffsdatum: 2025-12-13

[15] LogRocket: *Using Edge Functions in Supabase: A Complete Guide.* 2024. – URL https://blog.logrocket.com/using-edge-functions-supabase-complete-guide/. – Zugriffsdatum: 2025-12-26. – Praxisleitfaden zu Architektur und Einsatzszenarien von Supabase Edge Functions

[16] Make.com: *Google Docs Integration — Workflow Automation.* 2024. – URL https://www.make.com/en/integrations/auto-content-api/google-docs. – Zugriffsdatum: 2025-12-26

[17] Make.com: *Google Docs Integration | Workflow Automation*. 2024. – URL https://www.make.com/en/integrations/google-docs. – Zugriffsdatum: 2025-12-27. – Dokumentation der Module zur Template-Erstellung, -Befüllung und zum Export

[18] Make.com: *Airtable Modules — Search records*. 2025. – URL https://apps.make.com/airtable-modules. – Zugriffsdatum: 2025-12-24. – Abruf am 24.12.2025

[19] Make.com: *WhatsApp Business Cloud — Modules Documentation*. 2025. – URL https://apps.make.com/whatsapp-business-cloud-modules. – Zugriffsdatum: 2025-12-24. – Abruf am 24.12.2025

[20] Make.com Community: *How To Connect Make.com to Google Drive (2025 Tutorial)*. 2025. – URL https://www.youtube.com/watch?v=AiUT9cOCmVU. – Zugriffsdatum: 2025-12-26. – YouTube-Tutorial zu Google-Drive-Anbindung

[21] Media, Seibert: *Qualität, funktionale und nichtfunktionale Anforderungen in der Softwareentwicklung*. 2018. – URL https://seibert.group/blog/2018/05/14/qualitaet-funktionale-und-nichtfunktionale-anforderungen-in-der-software-entwicklung/. – Zugriffsdatum: 2025-12-13

[22] Pleo: *Automatisierte Rechnungsverarbeitung: Warum sie durch die E-Rechnung immer wichtiger wird*. 2024. – URL https://blog.pleo.io/de/automatisierte-rechnungsverarbeitung. – Zugriffsdatum: 2025-12-13

[23] 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. – Zugriffsdatum: 2025-12-13

[24] Supabase: *Edge Functions*. 2025. – URL https://supabase.com/docs/guides/functions. – Zugriffsdatum: 2025-12-26. – Dokumentation zu serverlosen, global verteilten Edge Functions

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
LaTeX	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments

Erklärung zur selbständigen Bearbeitung

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

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

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

Ort	Datum	Unterschrift im Original