THM | CAMPUS GIESSEN | MNI
TECHNISCHE HOCHSCHULE MITTELHESSEN
Mathematik, Naturwissenschaften und Informatik

Bachelorarbeit

Konzeptionierung und Implementierung einer Pivot-Tabellen-Komponente in einem CRM-System

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

vorgelegt dem
Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen

vorgelegt von

Marcel Frank Kucera

im September 2025

Referent: Sebastian Süß, M.Sc.
Korreferent: Prof. Dr. Steffen Vaupel

Inhaltsverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort Datum Unterschrift

TODO: KI Erklärung

TODO: Zusammenfassung

TODO: Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen datengetriebenen Geschäftswelt spielen Customer Relationship Management (CRM)-Systeme eine zentrale Rolle bei der Kundengewinnung, -bindung sowie beim Aufbau und der Pflege langfristiger Kundenbeziehungen [1, S. 342]. CRM-Systeme erfassen und verarbeiten dabei eine Vielzahl von Informationen – von Kundenkontakten und Interaktionen über Verkaufschancen bis hin zu Serviceanfragen und -tickets.

Doch die bloße Verfügbarkeit großer Datenmengen führt nicht automatisch zu besseren Entscheidungen. Ohne benutzerfreundliche und leistungsfähige Analysetools bleibt das Potenzial dieser Daten weitgehend ungenutzt. Besonders Vertriebs- und Marketingteams stehen regelmäßig vor analytischen Fragestellungen wie: „In welchen Regionen sind unsere Produkte im letzten Jahr unterdurchschnittlich gelaufen?“ oder „Wie hat sich der Umsatz nach Vertriebsregionen in den letzten drei Jahren entwickelt?“ Solche Fragen zielen darauf ab, Entwicklungen nachzuvollziehen und aufkommende Trends frühzeitig zu erkennen, um fundierte strategische Entscheidungen treffen zu können [1, S. 345].

Pivot-Tabellen sind ein bewährtes Mittel, um genau solche Fragestellungen interaktiv und effizient zu beantworten. Sie ermöglichen es den Nutzern, komplexe Datenbestände flexibel und intuitiv zu gruppieren, zu aggregieren und auszuwerten. Beispielsweise können Verkaufszahlen nach Vertriebsregion, Jahr oder Kundensegment gegliedert und analysiert werden. Dabei lassen sich zentrale Kennzahlen wie Umsätze oder Stückzahlen mithilfe von verschiedenen Aggregationsfunktionen (Summe, Durchschnitt, etc.) berechnen und übersichtlich in Tabellenform darstellen. Ergänzend dazu bieten weitere mögliche Visualisierungsmöglichkeiten der Pivot-Tabelle wie Balken-, Linien- oder Kreisdiagramme eine anschauliche Aufbereitung der Ergebnisse, wodurch sich Muster und Entwicklungen schneller erfassen und kommunizieren lassen.

Die Implementierung einer Pivot-Komponente direkt im CRM-System bietet aus technischer und organisatorischer Sicht mehrere Vorteile. Insbesondere ermöglicht sie die Nutzung der Datensätze an zentraler Stelle im System, wo die Daten bereits erfasst und gepflegt werden. Dadurch entfällt die Notwendigkeit, die Daten in externe Anwendun-

gen zu exportieren und dort weiterzuverarbeiten, was mit zusätzlichem Aufwand und potenziellen Inkonsistenzen verbunden sein kann.

Darüber hinaus erleichtert die Integration einer solchen Komponente die Einbettung analytischer Methoden in bestehende Arbeitsabläufe und Prozesse. Anwender können direkt innerhalb der vertrauten CRM-Systemumgebung Auswertungen interaktiv erstellen und nutzen, ohne zwischen verschiedener Software wechseln zu müssen. Auf diese Weise kann die Auswertung vorhandener Daten, sowie die darauf aufbauende Entscheidungsfindung, unterstützt werden.

1.2 Zielsetzung

Das Ziel dieser Bachelorarbeit ist die Konzeption und Umsetzung einer Pivot-Tabellen-Komponente im Kontext des CURSOR-CRM-Systems (https://www.cursor.de). Es stellt über sogenannte Infoboards eine Oberfläche bereit, auf der verschiedene Kacheln platziert werden können. Diese Kacheln ermöglichen unter anderem Datenvisualisierungen und -analysen. Die Komponenten lassen sich flexibel in das Infoboard einfügen und können dabei auf den Kontext der jeweiligen Umgebung als Datenquelle zurückgreifen.

Die entwickelte Komponente soll als eine „Infoboard-Kachel“ in die bestehende Systemarchitektur integriert werden und somit Zugriff auf die vorhandenen Datenquellen, wie Suchen, Masken und Unterbereiche, haben, sowie eine einheitliche Oberfläche zur Konfiguration der Komponente anbieten.

Im System existiert bereits eine Pivot-Tabellen-Komponente für Infoboards. Diese basiert jedoch auf veralteten Technologien, die im kommenden Jahr aus dem System entfernt werden sollen. Darüber hinaus hat die derzeitige Version nur begrenzte Anpassungsmöglichkeiten an kundenspezifische Anforderungen. Aus diesen Gründen ergibt sich der Bedarf für die Entwicklung einer aktualisierten, flexibler einsetzbaren und zukunftssicheren Version.

Die Integration in bestehende Arbeitsprozesse sowie CRM-Funktionalitäten stehen bei der Umsetzung im Vordergrund. Die Komponente soll so gestaltet sein, dass sie für den Nutzer einfach konfigurierbar und intuitiv bedienbar ist. Zudem soll die Komponente performant sein, um auch bei großen Datenmengen eine flüssige Benutzererfahrung zu bieten. Des Weiteren soll die entwickelte Komponente die firmeninternen Entwicklungsstandards einhalten sowie bestehenden Code wiederverwenden, damit sie vom Entwicklungsteam auch in Zukunft einfach gewartet werden kann.

1.3 Methodik

TODO: dashier nochmal angucken

Die Arbeit wird in mehrere aufeinanderfolgende Phasen gegliedert.

Zu Beginn erfolgt eine Anforderungsanalyse, auf deren Grundlage die Konzeption der Pivot-Komponente entwickelt wird. Im Anschluss daran wird ein technischer Entwurf erstellt, bei dem auch geeignete externe Bibliotheken hinsichtlich ihrer Eignung und Kompatibilität untersucht werden.

Die Implementierung der Komponente erfolgt iterativ. In einer ersten Phase wird ein unabhängiger Prototyp entwickelt, der zunächst losgelöst vom CURSOR-CRM-System funktioniert. Nach Fertigstellung eines funktionsfähigen Prototyps erfolgt die Integration in die CRM-Umgebung, einschließlich der Implementierung erforderlicher Schnittstellen. Dabei werden bestehende Systemlogiken und Bibliotheken, soweit möglich, übernommen und wiederverwendet.

Parallel zur technischen Entwicklung werden fortlaufend Usability-Tests durchgeführt, um frühzeitig Rückmeldungen zur Benutzerfreundlichkeit zu erhalten und diese in den Entwicklungsprozess einfließen zu lassen.

2 Kontext

2.1 Pivot-Tabellen

Pivot-Tabellen sind ein hilfreiches Werkzeug zur Datenanalyse und zum Reporting. Häufig werden sie im Zusammenhang mit Tabellenkalkulationssoftware wie Microsoft Excel verwendet, wo sie es ermöglichen, große Datenmengen interaktiv zu gruppieren, zu filtern, zu aggregieren und zu visualisieren.

Das Grundprinzip besteht darin, Datenfelder in Zeilen- und Spaltenachsen anzuordnen. Dabei werden numerische Werte aus einem anderem Feld der jeweiligen Daten an den Kreuzungen aggregiert, beispielsweise als Summe oder Durchschnitt. Über Filterfunktionen lassen sich spezifische Ausschnitte der Daten betrachten. So kann beispielsweise ausgewertet werden, wie viele Produkte ein bestimmter Vertriebsmitarbeiter in einem Monat verkauft hat.

Moderne Pivot-Tabellen ergänzen die tabellarische Ansicht häufig durch Diagramme wie Balken- oder Kreisdiagramme, um Muster und Trends besser sichtbar zu machen. Insgesamt abstrahieren Pivot-Tabellen SQL-ähnliche Operationen in einer benutzerfreundlichen Oberfläche, sodass auch fachliche Anwender komplexe Datenanalysen ohne Programmier- oder Datenbankkenntnisse durchführen können.

	Monat	April	Januar	Juli	Oktober	Totals
Verkäufer						
Dieter		220.00	159.00	230.00	215.00	**824.00**
Julia		230.00	200.00	270.00	240.00	**940.00**
Sabine		210.00	180.00	235.00	250.00	**875.00**
Thomas		180.00	140.00	210.00	170.00	**700.00**
	Totals	**840.00**	**679.00**	**945.00**	**875.00**	**3,339.00**

Abbildung 1: Beispiel einer Pivottabelle mit Summierung der Auftragswerte [2]

2.2 CURSOR-CRM

Das CURSOR-CRM ist ein Customer Relationship Management System, das Unternehmen bei der strukturierten Erfassung, Verwaltung und Auswertung ihrer Kunden- und

Geschäftsdaten unterstützt. Eine zentrale Funktion des Systems besteht in der Speicherung verschiedenartiger Daten, welche Entitäten genannt werden.

Entitäten beschreiben eine Klasse gleichartiger Datenobjekte. Sie definieren die Felder und Eigenschaften der jeweiligen Datensätze, wie zum Beispiel Geschäftspartner, Mitarbeitende, Angebote oder Aktivitäten. Jeder Datensatz gehört zu einer bestimmten Entität und kann mit anderen Datensätzen verknüpft sein. Die Anzeige und Bearbeitung der einzelnen Datensätze erfolgt über sogenannte Masken, in denen die zugehörigen Felder benutzerfreundlich dargestellt werden. Diese Masken sind anpassbar und lassen sich durch Scripting erweitern, um individuelle Anforderungen und Prozesse abzubilden.

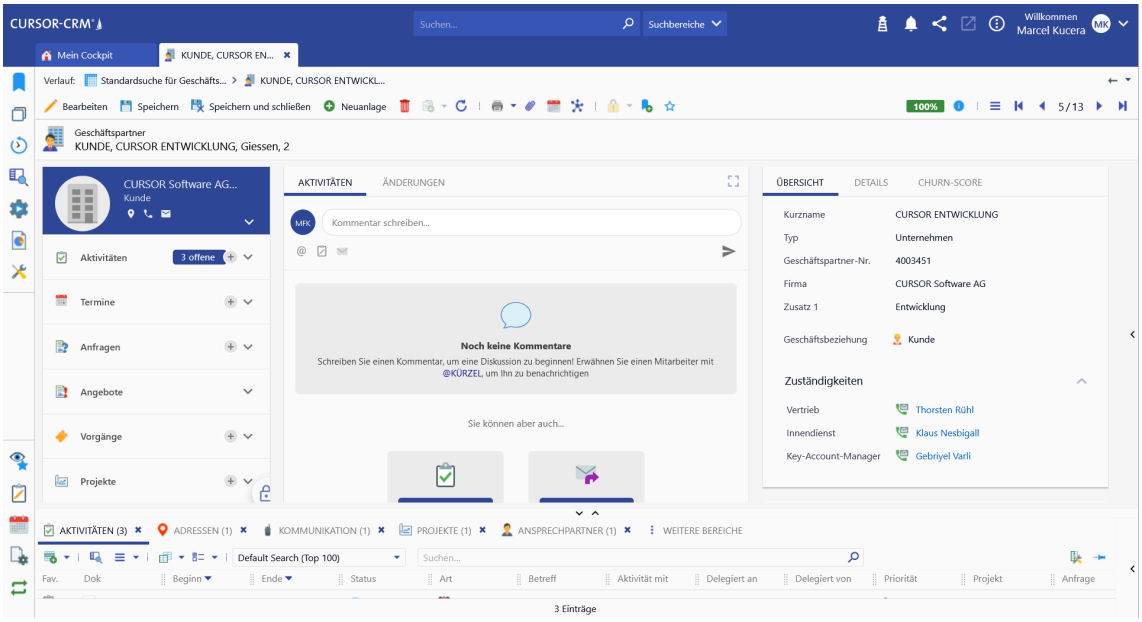

Abbildung 2: Screenshot des CURSOR-CRM auf der Maske für einen Geschäftspartner

Das CURSOR-CRM lässt sich flexibel an individuelle Unternehmensanforderungen anpassen. Neue Entitäten können hinzugefügt und bestehende modifiziert werden. Über vielfältige Schnittstellen ist zudem eine Integration mit externen Systemen möglich, sodass ein reibungsloser Datenaustausch innerhalb bestehender IT-Landschaften gewährleistet ist.

Im CURSOR-CRM stehen sogenannte „Suchen“ zur Verfügung, mit denen Anwender gezielt Daten anhand bestimmter Kriterien auffinden können. Diese Suchen sind ein zentraler Bestandteil des Systems und ermöglichen es, nicht nur innerhalb einzelner Entitäten zu filtern, sondern auch über deren Beziehungen hinweg komplexe Datenabfragen durchzuführen. Nutzerinnen und Nutzer haben dabei die Möglichkeit, eigene

Suchen zu erstellen und individuell anzupassen. Zudem lässt sich festlegen, welche Felder in den Suchergebnissen angezeigt werden, sodass die Darstellung exakt auf die jeweiligen Anforderungen abgestimmt werden kann.

2.3 Infoboards und Kacheln im CURSOR-CRM

Infoboards im CURSOR CRM dienen als flexible Oberfläche zur Visualisierung und Strukturierung von Informationen. Sie können in unterschiedlichen Kontexten eingesetzt werden, etwa in Dashboards, innerhalb von Masken oder den Unterbereichen der Masken, wo die verknüpften Entitäten angezeigt werden. Auf diesen Infoboards lassen sich sogenannte Kacheln platzieren, die individuell konfigurierbar sind und unterschiedliche Funktionen übernehmen können.

Kacheln ermöglichen es, kontextbezogene Daten dynamisch zu verarbeiten und visuell aufzubereiten. Dabei greifen sie auf Informationen aus dem jeweiligen Anzeigekontext zu, verarbeiten diese und stellen sie in geeigneter Form für den Nutzer dar. Typische Anwendungsbeispiele sind ToDo-Listen, die offene Aktivitäten anzeigen, grafische Darstellungen wie Diagramme oder einfache tabellarische Auflistungen relevanter Datensätze. Die Anordnung der Kacheln auf einem Infoboard kann per Drag-and-Drop angepasst werden, wodurch sich individuelle Arbeitsoberflächen schnell und intuitiv gestalten lassen. Kacheln lassen sich über ein Konfigurationsmenü individuell anpassen, wobei neben allgemeinen Einstellungen wie Inhalt, Darstellung und Sortierung auch kachelspezifische Optionen zur Verfügung stehen.

2.4 Typescript

TypeScript ist eine von Microsoft entwickelte Programmiersprache, die eine typsichere Erweiterung von JavaScript darstellt. Der Quellcode wird mithilfe eines Transpilers in JavaScript übersetzt, sodass eine Ausführung in allen gängigen JavaScript-Umgebungen, und somit auch dem Browser, möglich ist. Die Sprache unterstützt verschiedene Programmierparadigmen, darunter objektorientierte und funktionale Ansätze. Ein zentrales Merkmal von TypeScript ist die statische Typisierung. Sie erlaubt es, Typfehler bereits zur Entwicklungszeit zu erkennen und zu vermeiden, wodurch potenzielle Laufzeitfehler frühzeitig identifiziert werden können.

2.5 React

React ist eine von Meta entwickelte JavaScript-Bibliothek zur Erstellung dynamischer Benutzeroberflächen. Sie basiert auf einer komponentenbasierten Architektur, bei der wiederverwendbare UI-Elemente, sogenannte Komponenten, sowohl Struktur als auch Verhalten kapseln. Dies ermöglicht eine modulare und übersichtliche Entwicklung komplexer Webanwendungen.

Ein zentrales Konzept von React ist der Virtual DOM, eine abstrahierte Repräsentation des tatsächlichen DOM im Speicher. Änderungen am Anwendungszustand werden dort simuliert und durch einen Vergleichsalgorithmus gezielt und effizient auf den realen DOM übertragen. Dadurch wird die Performance bei der Darstellung dynamischer Inhalte deutlich verbessert.

React folgt einer deklarativen Programmierweise. Entwicklerinnen und Entwickler beschreiben den gewünschten UI-Zustand, während React sich um dessen Umsetzung kümmert. Zum Einsatz kommt dabei JSX, eine Syntaxerweiterung, die HTML-ähnliche Strukturen direkt in JavaScript integriert. Das erleichtert die Strukturierung und Wartung von Komponenten.

React ist als Bibliothek konzipiert und konzentriert sich hauptsächlich auf die Darstellung von Benutzeroberflächen. Durch diesen modularen Aufbau lässt sich React problemlos in bestehende Anwendungen integrieren, beispielsweise um einzelne UI-Komponenten schrittweise zu modernisieren oder interaktiver zu gestalten. Gleichzeitig besteht jedoch auch die Möglichkeit, mit Hilfe von darauf aufbauenden Frameworks komplette Anwendungen umzusetzen.

2.6 Kachel V1 zu V2

Im aktuellen System existieren zwei Versionen des Kachel-Systems, die sich hinsichtlich ihrer technologischen Grundlage unterscheiden. Die erste Version, welche als Kachel V1 bezeichnet wird, basiert auf klassischen Webtechnologien wie Vanilla JavaScript, HTML und CSS. Diese Variante ist funktional weiterhin im Einsatz, soll jedoch perspektivisch vollkommen durch die modernere Lösung Kachel V2 ersetzt werden.

Das neue Kachel V2 System basiert auf dem JavaScript-Framework React. Durch die Verwendung von React ist es möglich, auf eine bestehende Sammlung von wiederverwendbaren Komponenten innerhalb des CURSOR-CRM zuzugreifen. Dies ermöglicht

ein konsistenteres Erscheinungsbild zwischen der Systemoberfläche und den Kacheln und kann die Entwicklung und Wartung neuer Kacheln erleichtern. Darüber hinaus bietet diese zweite Version eine verbesserte Wart- und Erweiterbarkeit, da sie auf modernen Entwicklungsstandards aufbaut.

TODO: in erfahrung bringen warum genau gewechselt wird

3 Konzeptionierung

3.1 Betrachtung alte Pivot-Tabellen-Komponente

Die bisher im CURSOR-CRM eingesetzte Pivot-Tabelle, folgend als „Pivot-V1“ bezeichnet, zählt zu den funktional umfangreichsten Kacheln des Systems. Ähnlich wie Diagramm-Kacheln, KPI-Anzeigen oder die Suchergebnisliste unterstützt sie komplexere Analyse- und Auswertungsaufgaben basierend auf dem Infoboard-Kontext oder einer hinterlegten Suche.

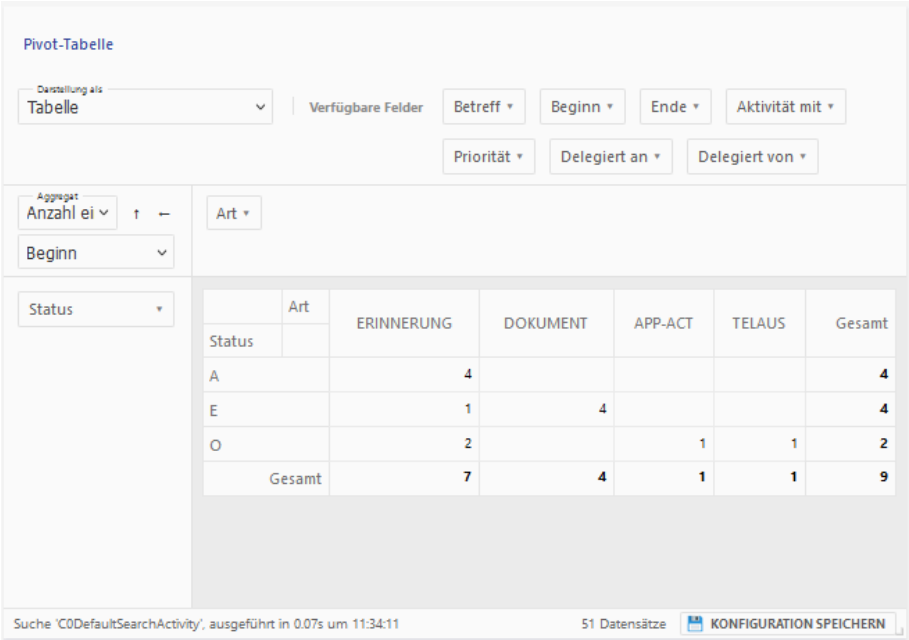

Abbildung 3: Pivot-V1 im CRM-System

Fundamental bietet die bestehende Komponente alle wesentlichen Pivot-Funktionen. Grundsätzlich ist die Komponente grob in zwei Teile, Konfiguration und Auswertung, unterteilt. Die Konfiguration bildet einen Rahmen um die Auswertung und stellt Dropdowns für die Art der Darstellung und Aggregation sowie eine Liste der für die Auswertung verfügbaren Felder bereit. Diese Felder kann man dann in die Zeilen und Spalten ziehen, um die Analyse nach diesen Feldern zu unterteilen. Zusätzlich kann man die Werte der Spalten und Zeilen sortieren

Für jedes verfügbare Feld kann man die betrachteten Werte über ein Dropdown eingrenzen. So kann man beispielsweise für eine Analyse von Angeboten nur die Abgeschlossenen in die Analyse miteinbeziehen. In dem Dropdown gibt ein Suchfeld, mit welchem man Werte suchen kann. So kann man einen gewünschten Wert schneller

finden und muss diesen nicht in der Liste von potentiell vielen Werten suchen. Ebenso gibt es Buttons, mit welchen man alle Werte für die Eingrenzung aus- oder abwählen kann.

Eine weitere Konfigurationsmöglichkeit ist die Auswahl der Aggregatsfunktion, mit welcher die Datensätze zusammengefasst werden sollen. Zu diesen Aggregatsfunktionen gehören unter Anderem Anzahl, Summe, Durchschnitt und Median. Wird eine Aggregatsfunktion ausgewählt, welche Werte zusammenfasst, wie beispielsweise Summe oder Durchschnitt, wird automatisch ein Dropdown angezeigt, in welchem man auswählen kann, welches Feld der Datensätze aggregiert werden soll.

Basierend auf diesen Konfigurationen wird automatisch die Auswertung und Visualisierung der Analyse in der unteren Rechten Ecke der Komponente angezeigt. Zusätzlich kann man die derzeitige Konfiguration mit einem Button in der unteren Leiste speichern, damit diese auch beim Neuladen der Kachel oder des Infoboards bestehen bleibt.

Abgesehen von der Konfiguration auf der Oberfläche der Pivot-Tabelle selbst, muss man ebenfalls weitere Konfigurationen in dem Kachel-Konfigurationsmenü vornehmen. Wenn man die Kachel auf einem Infoboard plaziert zeigt diese zuerst eine Fehlermeldung an, dass die Konfiguration fehlerhaft ist. Der Nutzer muss daraufhin das separate Konfigurationsmenü öffnen und dort mindestens die Datenquelle oder die hinterliegende Suche eintragen, wobei bei einer Suche der systeminterne Name verwendet werden muss. In diesem Menü kann man ebenfalls einstellen ob die Datumswerte nach Tag, Monat oder Jahr zusammengefasst werden sollen. Zudem kann man auswählen ob für Schlüssel der Schüsselname, die Beschreibung oder beides angezeigt werden soll.

Aus technischer Sicht basiert die alte Pivot-Kachel auf dem alten Kachel-V1-System und nutzt die inzwischen nicht mehr gepflegte Open-Source-Bibliothek „pivottable“[3], welche selbst auf der jQuery-Bibliothek basiert. Die neuste Version dieser Bibliothek ist v2.23.0, welche vor 6 Jahren veröffentlicht wurde [4]. Insbesondere das veraltete Kachel-V1-System, welches in zukünftigen Versionen entfernt werden soll, macht eine Neuentwicklung erforderlich, wobei die fehlende Weiterentwicklung der eingesetzten Bibliothek die zukünftige Wartbarkeit zusätzlich erschwert.

TODO: pm fragen, welche technischen gründe es noch geben könnte, um die kachel neu zu entwickeln

3.2 Ziele der neuen Implementierung

Die Neuentwicklung der Pivot-Kachel hat das grundlegende Ziel die Komponente in das Kachel-V2-System zu bringen, wodurch die alte, auf dem Kachel-V1-System basierende Komponente, aus dem CRM-System entfernt werden kann. Zusätzlich sollen die nicht mehr gewarteten Bibliotheken durch neue Bibliotheken ersetzt werden, welche mit React und somit dem Kachel-V2-System kompatibel sind.

Dabei sollen bevorzugt Bibliotheken eingesetzt werden, die bereits in anderen Bereichen des Systems genutzt werden. Eine reduzierte Anzahl verwendeter Bibliotheken senkt die Komplexität des Systems und erleichtert die Pflege der Abhängigkeiten. Insbesondere für die Visualisierung der Diagramme sollen die bewährten Bibliotheken aus der bestehenden Charts-Komponente wiederverwendet werden, um bei den Visualisierungen ein konsistentes Oberflächendesign sicherzustellen.

Ein weiteres Ziel der Neuentwicklung ist die stärkeren Nutzung von internen, wiederverwendbaren Komponenten, welche im Rahmen des Kachel-V2-Systems zur Verfügung stehen. Dadurch wird die Codebasis vereinheitlicht und gängige UI-Komponente systemweit konsistent gestaltet. Dies trägt ebenso zur Erhöhung der Wartbarkeit bei, da Anpassungen an zentralen Komponenten direkt an allen relevanten Stellen wirksam werden. Im Vergleich dazu bietet die Bibliothek der alten Pivot-V1-Komponente keine Möglichkeit die UI-Elemente anzupassen.

Gleichzeitig soll die Erweiterbarkeit der neuen Pivot-Kachel für die Entwicklung deutlich verbessert werden. Die Architektur der Komponente soll so gestaltet werden, dass zukünftige Funktionserweiterungen, wie beispielsweise die Integration neuer Visualisierungen, mit geringem Entwicklungsaufwand möglich sind.

Ein weiterer Schwerpunkt der Neuentwicklung und das Hauptmerkmal der Pivot-V2 ist die Verbesserung der Nutzerfreundlichkeit. Insbesondere die Abhängigkeit der Konfiguration über das Kachel-Konfigurationsmenü soll reduziert werden und die wichtigsten Konfigurationen, unter Anderem auch die Auswahl der Datenquelle, sollen innerhalb der Kachel verfügbar sein. In der Pivot-V1-Komponente wird ein Fehler angezeigt wenn die Kachel neu auf einem Infoboard hinzugefügt wird und noch keine Datenquelle konfiguriert ist.

Im Gegensatz dazu wird im Kachel-V2-System der sogenannte „Datasource-Editor“ angeboten. Dieser wird angezeigt, wenn die Kachel noch keine Datenquelle konfiguriert

hat. Der Datasource-Editor ist eine Maske, welche auf der Kachel selbst angezeigt wird. Sie ermöglicht es dem Nutzer eine Datenquelle auszuwählen. Verglichen zu dem Kachel-Konfigurationsmenü muss der Nutzer eine Suche nicht mit dem technischen Namen angeben, sondern kann diese über ein Feld anhand deren Anzeigenamen suchen und auswählen. Sobald der Nutzer eine Datenquelle ausgewählt hat, wird diese automatisch in der Konfiguration eingetragen und die Kachel ist funktionsbereit.

Darüber hinaus ist vorgesehen, den bisher separaten Button zum Speichern der derzeitigen Konfiguration der Pivot-Kachel zu entfernen. Stattdessen soll das Speichern zukünftig implizit erfolgen, sodass Konfigurationsänderungen automatisch übernommen werden und über ein Neuladen der Kachel hinweg persistieren.

Da die Pivot-V1-Komponente in einigen Kundensystemen weiterhin verwendet wird, soll eine automatisierte und möglichst reibungslose Migration bestehender Konfigurationen auf die neue Pivot-V2-Komponente implementiert werden. Ziel ist es, die bisher verwendeten Einstellungen weitgehend zu übernehmen, um den Umstieg für Anwender zu erleichtern und zusätzliche Konfigurationaufwände zu vermeiden.

3.3 Anforderungen

3.3.1 Nicht funktionale Anforderungen

TODO: Nicht funktionale Anforderungen

- usability
- performance
- wartbarkeit/erweiterbarkeit (insbesondere neue visualisierungen)
- zum system einheitliches design
- verwendung von standart komponenten
- minimaler einsatz von neuen bibliotheken

3.3.2 Anwendungsfälle

Zur Festlegung der funktionalen Anforderungen an die Pivot-Tabellen-Komponente wurden verschiedene Anwendungsfälle erarbeitet, die verdeutlichen, welche konkreten Nutzungsszenarien durch die Komponente unterstützt werden sollen. Die Formulierung und Strukturierung dieser Anwendungsfälle orientiert sich an der Methodik von Alistair Cockburn, wie sie in seinem Werk Writing Effective Use Cases beschrieben ist [5].

Eine Auswahl besonders relevanter Anwendungsfälle wird im Folgenden im Detail dargestellt.

- Anwendungsfall 1: Konfiguration der Datenquelle
- Anwendungsfall 2: Anzeigen der Verfügbaren Felder
- Anwendungsfall 3: Felder in die Spalten einfügen
- Anwendungsfall 4: Felder in die Zeilen einfügen
- Anwendungsfall 5: Auswahl einer Visualisierung
- Anwendungsfall 6: Auswahl einer Aggregationsfunktion
- Anwendungsfall 7: Auswahl eines Aggregationsfeldes
- Anwendungsfall 8: Aus- / Eingrenzen von Werten eines Feldes
- Anwendungsfall 9: Erzeugen der Auswertung
- Anwendungsfall 10: Mit Klick auf die Visualisierung eine Suche mit den relevanten Datensätzen öffnen
- Anwendungsfall 11: Datumsfelder nach Tag, Monat oder Jahr aggregieren
- Anwendungsfall 12: Schlüsselwerte als Schlüssel und/oder Beschreibung anzeigen

Anwendungsfall 1: Konfiguration der Datenquelle

Ziel: Kachel mit gewünschter Datenquelle konfigurieren
Vorbedingung: Keine
Nachbedingung: Kachel wurde mit einer Datenquelle konfiguriert und ist funktionsbereit
Nachbedingung im Fehlerfall: Fehlermeldung wird angezeigt und Konfiguration wird automatisch neugestartet
Auslöser: Kachel wird auf einem Infoboard hinzugefügt
Ablauf:

1. Kachel wird auf ein Infoboard gezogen
2. Der Nutzer gibt den Anzeigenamen einem Suchfeld ein
3. Es wird eine Liste von Suchen dem Nutzer vorgeschlagen
4. Der Nutzer wählt die gewünschte Suche aus
5. Die gewählte Suche wird in die Konfiguration übernommen
6. Konfigurationsmaske schließt sich und Kachel läd mit Inhalt

Erweiterung:

1a. Kachel wird auf Infoboard im Unterbereich plaziert
1a1. Datenquelle wird automatisch auf den Unterbereich gesetzt

Abbildung 4: Anwedungsfall Konfiguration der Datenquelle

TODO: Weitere Anwendungsfälle

3.4 Mockups

Die folgenden Mockups zeigen die geplante Benutzeroberfläche der neuen Pivot-Tabellen-Komponente und veranschaulichen die wesentlichen Interaktionselemente. Dazu wird der typische Arbeitsablauf beschrieben, den Nutzer bei der Konfiguration und Verwendung der Pivot-Tabelle durchlaufen.

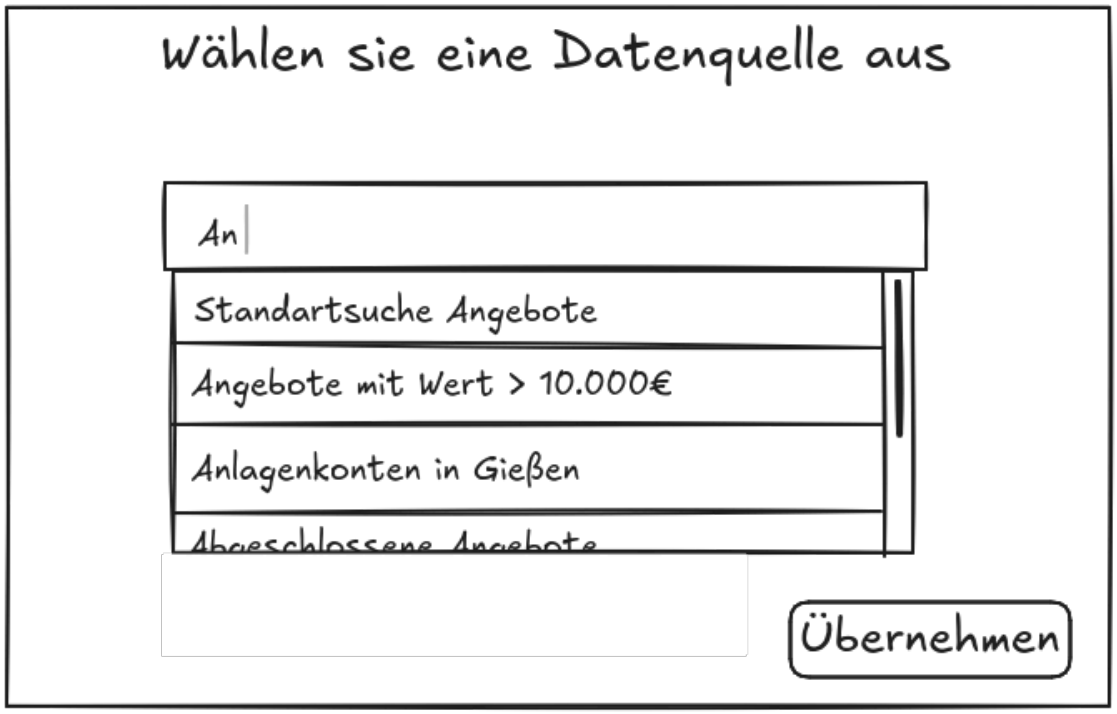

Abbildung 5: Mockup des Datasource-Selectors

Nachdem die Kachel auf einem Infoboard plaziert wurde, wird zunächst der Datasource-Selector angezeigt. In dieser ersten Ansicht wird der Nutzer dazu aufgefordert, eine Datenquelle bzw. Suche auszuwählen. Im Zentrum der Oberfläche befindet sich ein Suchfeld, über das die verfügbaren Datenquellen gesucht werden können. Sobald der Nutzer eine Eingabe tätigt, wird unter der Suchleiste eine Liste der passenden Suchergebnisse angezeigt. Der Nutzer kann ein Suchergebnis aus der Liste anklicken, um es zu übernehmen. Nach Auswahl der Datenquelle drückt der Nutzer auf den „Übernehmen“-Button. Dadurch wird gewählte Datenquelle in der Konfiguration der Kachel gespeichert. Nachdem die Datenquelle konfiguriert wurde, wird die Hauptansicht der Pivot-Tabelle automatisch geladen.

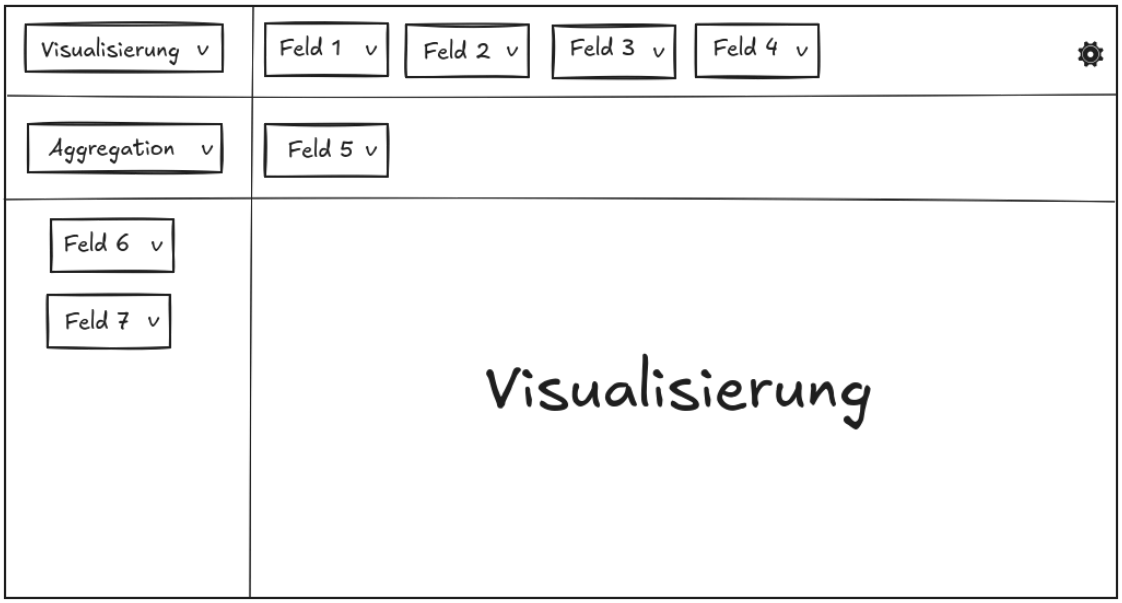

Abbildung 6: Mockup der Pivot-Tabelle

Die Hauptansicht der Pivot-Tabellen-Komponente ist in mehrere Bereiche unterteilt. Im oberen linken Bereich befindet sich die Auswahl der Visualisierungsart, darunter ist die Auswahl der Aggregationsfunktion. Im rechten oberen Bereich wird eine Liste verfügbarer Felder angezeigt, aus der Elemente per Drag-and-Drop in die Auswertung übernommen werden können. Die eigentliche Visualisierung nimmt den unteren rechten Bereich ein und bildet den größten Bereich der Oberfläche.

Die in der Analyse verwendeten Felder werden links und oberhalb der Visualisierung dargestellt. Die im linken Bereich platzierten Felder definieren die Zeilen beziehungsweise die Y-Achse der Auswertung, während die oberhalb angeordneten Felder den Spalten beziehungsweise der X-Achse zugeordnet sind. Felder können per Drag-and-Drop zwischen der Liste der verfügbaren Felder und den Bereichen für Zeilen und Spalten verschoben werden.

Darüber hinaus verfügen die einzelnen Felder über einen Button zum Ausklappen des jeweiligen Filterdialogs. Sobald sich eine Konfiguration ändert, wie etwa durch Auswahl eines anderen Visualisierungstyps, einer neuen Aggregationsfunktion oder das verschieben von Feldern, wird die Visualisierung automatisch aktualisiert. In der rechten oberen Ecke befindet sich ein Button, über den zusätzliche Konfigurationseinstellungen geöffnet werden können, beispielsweise zur Darstellung von Schlüsseln oder zur Aggregation von Datumswerten.

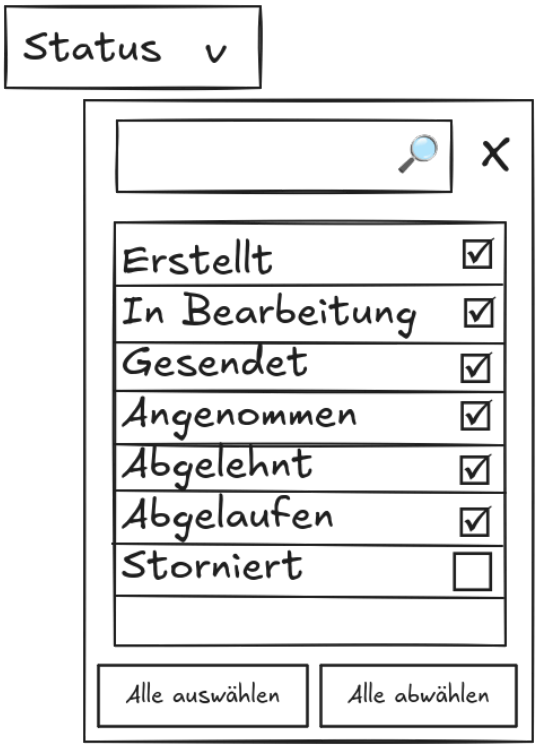

Abbildung 7: Mockup der Werteeingrenzung

Der Filterdialog ermöglicht eine gezielte Eingrenzung der in die Auswertung einbezogenen Werte der jeweiligen Felder. Im oberen Bereich befindet sich eine Suchleiste, über die die angezeigten Werte gefiltert werden können. Darunter wird eine Liste aller verfügbaren Werte angezeigt. Die Werte haben jeweils eine Checkbox, über die der Nutzer festlegen kann, ob der entsprechende Wert in die Auswertung aufgenommen werden soll.

Am unteren Rand des Dialogs befinden sich Schaltflächen, mit denen alle Werte gleichzeitig ausgewählt oder abgewählt werden können. Zum Schließen des Dialogs steht ein Button in der oberen rechten Ecke zur Verfügung. Alternativ kann das Dialogfenster auch durch einen Klick außerhalb des Popups geschlossen werden.

TODO: Mockup Tabelle? (eigentlich noch unklar wie, da eventuell hier ein Bibliothek benutzt wird)

4 Technischer Entwurf

4.1 Evaluation und Auswahl von verfügbaren Bibliotheken

- Eine Drag and Drop Library wird bereits im System verwendet. Diese wurde für diese Komponente auch verwendet.
- Kurz beschreiben was die macht, dass die eine einfachere version von react-dnd ist, aber für unsere anwendungsfälle reicht
- ein paar pivot libraries raussuchen. dabei eingehen, dass mui-x verwendet wird, aber an sich nicht den anforderungen entspricht
- der react port der bisher verwendeten version wir nichtmehr gewartet und ist mit neueren react versionen inkompatibel
  - hier auch ein bild der fehlermeldung einbinden
- visualisierungslibrary ist auch schon größtenteils vorgegeben (mui charts)
  - adapterkomponenten für die jeweiligen visualisierungen
  - erweiterbarkeit

4.1.1 react-pivottable

4.1.2 MUI-X Datagrid Pivottable

4.1.3 Hilfsbibliotheken

4.1.4 Fazit

4.2 Architektur

Die Architektur der Pivot-Tabellen-Komponente folgt einem modularen Aufbau, der eine klare Trennung zwischen Konfigurationslogik und der eigentlichen Darstellung der Auswertung vorsieht. Funktional gliedert sich die Komponente in zwei Hauptbereiche: den Konfigurationsrahmen, in dem Nutzerinnen und Nutzer die Auswertung definieren, und den Visualisierungsbereich, in dem die Daten in tabellarischer oder grafischer Form dargestellt werden.

Die Komponente selbst wird als eigenständige React-Komponente entwickelt, die systemweit verfügbar ist. In ihrer Grundstruktur ist sie zunächst unabhängig vom Kachel-V2-System konzipiert. Sie erhält sowohl die Konfiguration als auch die auszuwertenden Daten über definierte Parameter. Die eigentliche Integration in das Kachel-V2-System

erfolgt über eine dedizierte Kachel, welche die Kommunikation mit der Kachel-API übernimmt. Diese Kachel extrahiert die erforderlichen Informationen und übergibt sie in aufbereiteter Form an die Pivot-Komponente. Hintergrund dieser Aufteilung ist die Architektur des Kachel-V2-Systems, das bestimmte Einschränkungen aufweist – etwa die fehlende Unterstützung für Klassen oder eigenständige Funktionsdefinitionen. Durch die Trennung von Kachel und Komponente kann die Funktionalität der Pivot-Tabelle unabhängig von diesen Begrenzungen entwickelt werden.

Der Konfigurationsrahmen wird vollständig eigenentwickelt, um maximale Kontrolle über die Funktionalität, die Interaktion und das Nutzererlebnis zu gewährleisten. Für die Darstellung der Visualisierungen hingegen wird überwiegend auf bestehende Bibliotheken zurückgegriffen. Dies betrifft insbesondere die Diagrammdarstellung, für die auf bereits im System etablierte Bibliotheken zurückgegriffen wird, um Konsistenz und Wiederverwendbarkeit sicherzustellen. Die tabellarische Darstellung der Daten bildet eine Ausnahme, da diese spezifische Anforderungen an Layout, Interaktivität und Exportfunktionen stellt und daher weitgehend individuell umgesetzt wird.

Die gesamte Auswertung der Daten erfolgt clientseitig innerhalb der Komponente. Eine serverseitige Aggregation oder Vorverarbeitung findet nicht statt. Dieser Entwurfsansatz ist möglich, da die zugrunde liegenden Datenmengen durch den Server limitiert werden – standardmäßig auf maximal 20.000 Einträge. Die bisherige Nutzung und Performanzmessung im System zeigt, dass diese Datenmengen von modernen Browsern effizient verarbeitet werden können und eine clientseitige Umsetzung unter Berücksichtigung dieser Begrenzung als angemessen und performant einzustufen ist.

4.2.1 Komponenten

- server
- kachel
- Pivot-Tabellen-Komponente
- klasse zur datenverarbeitung
- feld listen (verfügbare felder, spalten, zeilen)
- felder haben filter
- visualisierungsadapter

4.2.2 datenverarbeitung

- hier weiß ich nicht so 100%, ob das wirklich benötigt ist, oder ob alle visualisierungslibraries mir hier die arbeit abnehmen
- hier werde ich zumindest mal die filter und die selektion der felder bearbeiten müssen
- hier könnte ich noch daten modell machen bzw. welchen zustand die kachel haben soll.
  - felder in Zeilen
  - felder in Spalten
  - ausgewählte visualisierung
  - ausgewählte Aggregationsfunktion
  - ausgewähltes feld für Aggregationsfunktion
  - ausgewählte filter
  - konfuguration (anzeige schlüsselfelder, aggragation von datumswerten)

4.2.3 Visualisierungsadapter

5 Implementierung

5.1 Freistehende Entwicklung

- prototypen
- einrichtung vite react
- bilder von dem jetzigen stand außerhalb des crm-systems
- entwicklung des rahmens
- entwicklung der klasse für zentrale datenverarbeitung
- entwicklung der tabelle
- entwicklung der adapter

5.2 Integration in das CRM-System

- hier auch genau erläutern wie die kacheln eingebunden werden
  - auch auf die verbindung von system, komponente im standard, bis hin zur kachel gehen
- zuerst eine einbindung in das system
- dann integration in eine kachel
- bei den kacheln auch die kachel api im detail erleutern
- bilder von dem prototypen
- ab hier auch regelmäßig stand checken gegenüber product management und entwicklung

5.3 Vergleich Tabellenkomponente MUI-X zu Eigenentwicklung

- hier nochmal erwähnen, dass die pivot-komponente erst während der entwicklung veröffentlicht wurde

5.4 Tests

6 Evaluation

6.1 Usability-Tests

6.2 Bewertung

7 Fazit und Ausblick

- gut
- erfüllt allen anforderungen

Bibliographie

[1] W. Becker *u. a.*, Hrsg., *Geschäftsmodelle in der digitalen Welt: Strategien, Prozesse und Praxiserfahrungen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. doi: 10.1007/978-3-658-22129-4.

[2] „React-Pivottable“. Zugegriffen: 12. Juli 2025. [Online]. Verfügbar unter: https://react-pivottable.js.org/

[3] N. Kruchten, „Nicolaskruchten/Pivottable“. Zugegriffen: 16. Juli 2025. [Online]. Verfügbar unter: https://github.com/nicolaskruchten/pivottable

[4] „Release Version 2.23.0 · Nicolaskruchten/Pivottable“. Zugegriffen: 27. Juli 2025. [Online]. Verfügbar unter: https://github.com/nicolaskruchten/pivottable/releases/tag/v2.23.0

[5] A. Cockburn, *Writing Effective Use Cases*, 24. print. in The Agile Software Development Series. Boston: Addison-Wesley, 2012.