Abschnitt 1: Einleitung

Die Kundenverwaltungsanwendung ist ein webbasiertes System, das die Verwaltung von Kundendaten vereinfacht. Es ermöglicht den Benutzern, wesentliche CRUD (Create, Read, Update, Delete)-Operationen an Kundenakten auszuführen. Die Anwendung bietet folgende Hauptfunktionen:

- Neue Kunden mit detaillierten Informationen, einschließlich Name, Kontaktdaten und Adresse, hinzufügen.
- Nach vorhandenen Kunden suchen, z. B. nach Name oder E-Mail.
- Kundendaten bearbeiten und aktualisieren.
- Kundenakten löschen, wobei alle Operationen eine persistente Datenspeicherung sicherstellen. Dieses Projekt dient als effizientes Tool für Unternehmen, die ihre Kundendatenbank systematisch pflegen und verwalten möchten.

Verwendete Technologien Für die Implementierung der Anwendung wurden folgende Technologien verwendet:

- **Frontend:** Entwickelt mit Vue 3, einem progressiven JavaScript-Framework, zusammen mit Vuetify für eine benutzerfreundliche und responsive Benutzeroberfläche. Wichtige Abhängigkeiten sind:
    - axios für HTTP-Anfragen.
    - vue-router für das Routing zwischen den Ansichten.
    - vuetify für vorgefertigte UI-Komponenten.
- **Backend:** Entwickelt mit Laravel, einem leistungsstarken PHP-Framework, das für seine elegante Syntax und entwicklerfreundliche Funktionen bekannt ist. Wichtige Abhängigkeiten sind:
    - cors für sichere Cross-Origin-Anfragen.
    - axios für die Kommunikation zwischen Frontend und Backend.
    - Vite als Build-Tool zur Optimierung von Assets.

Diese Anwendung demonstriert robuste Designprinzipien zur Trennung der Anliegen der Client- und Server-Ebenen und gewährleistet eine nahtlose Integration.

Abschnitt 2: Einrichtung und Installation

Um die Kundenverwaltungsanwendung auf Ihrem lokalen Rechner einzurichten und auszuführen, folgen Sie den untenstehenden Schritten. Diese Anleitung setzt Grundkenntnisse in Entwicklungswerkzeugen und -umgebungen voraus.

Voraussetzungen Stellen Sie sicher, dass folgende Software auf Ihrem System installiert ist:

- **Node.js:** Version 16 oder höher.
- **npm (Node Package Manager):** Wird mit der Node.js-Installation mitgeliefert.
- **PHP:** Version 8 oder höher zur Unterstützung von Laravel.
- **Composer:** Ein Abhängigkeitsmanager für PHP-Projekte.
- **Git:** Zum Klonen des Repositories.

Repository Klonen

1. Öffnen Sie ein Terminal oder eine Eingabeaufforderung.
2. Klonen Sie das Repository: `git clone <repository-url>`
3. Navigieren Sie in das Projektverzeichnis: `cd <repository-folder>`

Frontend Einrichtung

1. Navigieren Sie in das Frontend-Verzeichnis: `cd frontend`
2. Installieren Sie die erforderlichen Abhängigkeiten: `npm install`
3. Starten Sie den Entwicklungsserver: `npm run dev` Das Frontend ist unter http://localhost:8080 oder der im Terminal angezeigten Netzwerk-URL erreichbar.

Backend Einrichtung

1. Navigieren Sie in das Backend-Verzeichnis: `cd backend`
2. Installieren Sie Abhängigkeiten mit Composer: `composer install`
3. Richten Sie die Umgebung ein: Konfigurieren Sie die Datenbank und andere Umgebungsvariablen in der `.env`-Datei.
4. Generieren Sie den Anwendungsschlüssel: `php artisan key:generate`
5. Führen Sie die Datenbankmigrationen aus: `php artisan migrate`
6. Starten Sie den Entwicklungsserver: `php artisan serve` Das Backend ist unter http://127.0.0.1:8000 oder der angegebenen lokalen Serveradresse erreichbar.

Weitere Hinweise Stellen Sie sicher, dass sowohl der Frontend- als auch der Backend-Server laufen, damit die Anwendung korrekt funktioniert. Falls Cross-Origin-Probleme auftreten, prüfen Sie, ob CORS im Backend korrekt konfiguriert ist.

Abschnitt 3: Architektur

Die Architektur der Kundenverwaltungsanwendung folgt einem standardmäßigen Client-Server-Modell mit einem Vue 3 Frontend und einem Laravel Backend. Das Frontend ist

für die Benutzeroberfläche und Interaktion verantwortlich, während das Backend die Datenverwaltung und CRUD-Operationen über API-Endpunkte übernimmt.

Die Kommunikation zwischen Frontend und Backend erfolgt hauptsächlich durch HTTP-Anfragen, wobei Axios verwendet wird, um API-Endpunkte zum Abrufen, Erstellen, Aktualisieren und Löschen von Kundendaten anzusprechen. Für die MySQL-Datenbank im Backend wird XAMPP genutzt.

Frontend-Architektur Das Frontend ist so strukturiert, dass die Trennung der Anliegen beibehalten wird, wodurch der Code organisiert und modular bleibt:

- `src/assets/images`: Beinhaltet Assets wie Logos, die in der gesamten App verwendet werden.
- `src/assets/components/common`: Beinhaltet wiederverwendbare, modulare Komponenten wie Kopf- und Fußzeilen.
- `src/assets/components/pages`: Beinhaltet Komponenten, die mit den zentralen CRUD-Operationen für die Kundenverwaltung verbunden sind (z. B. Seiten zum Hinzufügen, Bearbeiten oder Anzeigen von Kunden).
- `src/assets/plugins/vuetify.js`: Konfiguriert Vuetify und verwaltet UI-Komponenten für ein responsives und elegantes Design.
- `src/assets/router/index.js`: Verwaltet das Routing innerhalb der Anwendung und ordnet Routen Ansichten oder Seiten zu.
- `src/App.vue`: Die Haupt-App-Komponente, die als Wurzel des Frontends dient und die primäre Layoutstruktur enthält.
- `README.md`: Beinhaltet notwendige Projektdokumentation, Installationsschritte und grundlegende Anweisungen.

Backend-Architektur Das Backend, entwickelt mit Laravel, folgt dem Model-View-Controller (MVC)-Muster, das eine saubere und wartbare Struktur gewährleistet:

- `backend/app/Http/Controllers/KundeController.php`: Verwaltet CRUD-Operationen für Kunden. Es verarbeitet Anfragen und gibt Antworten zurück, um sicherzustellen, dass die Daten korrekt manipuliert werden.
- `backend/app/Models/Kunde.php`: Das Kunden-Modell interagiert mit der Datenbank, stellt die Struktur der Kundendaten dar und bietet Methoden zum Abfragen von Daten.
- `backend/database/migrations/2025_01_13_231755_create_kunden_table.php`: Definiert das Schema für die „kunden“-Tabelle in der Datenbank.
- `backend/routes/api.php`: Definiert die API-Routen, die das Frontend mit den entsprechenden Controller-Methoden für die Kundenverwaltung verbinden.

Datenbank (XAMPP) XAMPP wird verwendet, um die MySQL-Datenbank zu verwalten und bietet eine benutzerfreundliche lokale Entwicklungsumgebung für das Backend.

Kommunikation zwischen Frontend und Backend Das Frontend kommuniziert mit dem Backend über API-Endpunkte unter Verwendung von Axios. Diese Endpunkte verwalten:

- Abrufen von Kundendaten.
- Senden neuer Kundenakten an die Datenbank.
- Aktualisieren und Löschen von Kundenakten im System.

Abschnitt 4: Funktionen

Die Kundenverwaltungsanwendung bietet wesentliche CRUD-Funktionen (Create, Read, Update, Delete) zur effizienten Verwaltung von Kundendaten. Das System ermöglicht es dem Benutzer, Kundeninformationen einzugeben, abzurufen, zu ändern und zu löschen. Das Frontend interagiert über API-Aufrufe mit dem Backend, um diese Aktionen auszuführen.

Wichtige Funktionen

- **Neuen Kunden hinzufügen**
  Der Benutzer kann einen neuen Kunden erstellen, indem er folgende Details bereitstellt:
    - Nachname
    - Vorname
    - E-Mail
    - Telefonnummer
    - Adresse (Straße + Nummer, Postleitzahl, Stadt)
- Nach der Eingabe werden die Kundendaten in der Datenbank gespeichert und eine eindeutige Kunden-ID zugewiesen.

- **Existierenden Kunden suchen**
  Benutzer können einen Kunden mit folgenden Parametern suchen:
    - Nachname und Vorname
    - E-Mail-Adresse
    - Kunden-ID
- Die Suchergebnisse werden in einer Liste angezeigt, sodass der Benutzer einen gewünschten Kunden auswählen kann, um die Details anzusehen oder zu ändern.

- **Kundendaten anzeigen**
  Beim Auswählen eines Kunden aus den Suchergebnissen werden folgende

Informationen angezeigt:

  - Nachname
  - Vorname
  - E-Mail
  - Telefonnummer
  - Vollständige Adresse (Straße + Nummer, PLZ, Stadt)
- **Kundendaten bearbeiten**
  Benutzer können folgende Details eines bestehenden Kunden ändern:

  - Nachname
  - Vorname
  - E-Mail
  - Telefonnummer
  - Adresse (Straße + Nummer, PLZ, Stadt)
- Nach der Aktualisierung werden die Änderungen in der MySQL-Datenbank gespeichert.

- **Kunden löschen**
  Benutzer haben die Möglichkeit, die Informationen eines bestehenden Kunden zu löschen. Die Löschung ist permanent und die Daten werden aus der Datenbank entfernt.

Benutzeroberfläche Das Frontend, entwickelt mit Vue 3 und Vuetify, bietet eine einfache und benutzerfreundliche Oberfläche:

- Klar strukturierte Formulare zum Eingeben von Kundendaten.
- Eine responsive, benutzerfreundliche Such- und Auswahlmöglichkeit für die Anzeige von Kunden.
- Formulare zur Aktualisierung oder Löschung von Kundendaten mit entsprechender UI-Rückmeldung (Erfolgs-/Fehlermeldungen).

API-Kommunikation Axios wird für HTTP-Anfragen an das Backend verwendet, um Kundenakten zu erstellen, abzurufen, zu aktualisieren oder zu löschen.

Abschnitt 5: Datenbankdesign

Die Anwendung verwendet eine MySQL-Datenbank, die über XAMPP verwaltet wird, um Kundendaten dauerhaft zu speichern. Das Datenbankdesign folgt einer einfachen Struktur, um alle notwendigen Kundendaten für eine einfache Abfrage, Aktualisierung und Löschung zu halten.

Tabelle Kunden:

- **Tabellenname:** kunden
- **Spalten:**
    - ID: Primärschlüssel, wird automatisch inkrementiert.
    - Nachname: Speichert den Nachnamen des Kunden.
    - Vorname: Speichert den Vornamen des Kunden.
    - E-Mail: Speichert die E-Mail-Adresse des Kunden.
    - Telefonnummer: Speichert die Telefonnummer des Kunden.
    - Adresse (Straße + Nummer, PLZ, Stadt): Speichert die vollständige Adresse.

Datenbankmigration Die Migrationsdatei `2025_01_13_231755_create_kunden_table.php` definiert die Struktur der "kunden"-Tabelle.

Abschnitt 6: APIs

Das Backend stellt eine Reihe von RESTful API-Endpunkten zur Verfügung, um Kundenverwaltungsoperationen (CRUD) durchzuführen.

Wichtige API-Endpunkte:

- **POST /api/kunden:** Erstellen eines neuen Kunden.
- **GET /api/kunden/{id}:** Abrufen von Kundendetails.
- **PUT /api/kunden/{id}:** Aktualisieren von Kundendaten.
- **DELETE /api/kunden/{id}:** Löschen eines Kunden.

Abschnitt 7: Fehlerbehandlung

Die Fehlerbehandlung ist implementiert, um einen reibungslosen Betrieb der Anwendung zu gewährleisten, indem Fehler während der CRUD-Vorgänge abgefangen und entsprechend darauf reagiert wird.

Frontend-Fehlerbehandlung

- **Axios**-Interceptors werden verwendet, um Fehler von API-Anfragen abzufangen, wie z.B. das Erstellen, Aktualisieren oder Löschen von Kunden.
- **Benachrichtigungen** oder **Alert-Dialoge** zeigen Fehler wie falsche Eingaben an.
  Häufig behandelte Fehlertypen umfassen:
  400 Bad Request (z.B. ungültige Eingabe).

404 Not Found (z.B. ungültige URL).
500 Internal Server Error (z.B. Serverprobleme).

Backend-Fehlerbehandlung

Laravel verwendet sein System zur Ausnahmebehandlung, um Fehler zu verwalten und korrekte Antworten zurückzugeben:

- **200 OK** für erfolgreiche Vorgänge.
- **400 Bad Request** für Fehler bei der Client-seitigen Validierung.
- **404 Not Found**, wenn ein angeforderter Kunde oder eine Ressource nicht gefunden wird.
- **500 Internal Server Error** für Server-seitige Fehler.

Validierung

- Die Eingabvalidierung wird sowohl im Frontend als auch im Backend angewendet, um das richtige Datenformat und die Vollständigkeit vor der weiteren Verarbeitung zu gewährleisten.

Abschnitt 8: Fazit

Diese Kundenverwaltungsanwendung bietet Benutzern eine einfache Oberfläche zum Durchführen von CRUD-Operationen, mit der Möglichkeit, Kundeninformationen zu erstellen, zu suchen, zu aktualisieren und zu löschen. Die Anwendung ist mit **Vue 3** für das Frontend und **Laravel** für das Backend entwickelt, wobei **MySQL** zur Datenspeicherung verwendet wird.

Zukünftige Verbesserungen

- **Benutzerauthentifizierung**: Implementierung einer Benutzerauthentifizierung für sicheren Zugriff auf Kundendaten.

Schlussgedanken

Dieses Projekt zeigt, wie Frontend- und Backend-Technologien zusammenarbeiten können, um eine effiziente und benutzerfreundliche Anwendung zu schaffen, die den Projektanforderungen entspricht und bei Bedarf weiter ausgebaut werden kann.