Dokumentation zur schulischen Projektarbeit

Fachinformatiker für Anwendungsentwicklung

Entwickeln einer Software für Gebäudemanagement

Schüler:
Ahmed Yasin
Poppenrade 13
24148 Kiel

Prüfer:
Dennis Clausen
BBZ Rendsburg-Eckernförde
Kieler Straße 30
24768 Rendsburg

Inhaltsverzeichnis

1. Anforderungsanalyse

1.1. Funktionale Anforderungen

- **Gebäudeverwaltung:** Erfassen, Bearbeiten, Löschen von Gebäuden (Straße, Nr., Ort, PLZ).
- **Mieterverwaltung:** Anlegen, Bearbeiten, Entfernen von Mietern (Name, Geburtsdatum/-ort, Geschlecht).
- **Wohnungsverwaltung:** Verknüpfung mit Gebäuden/Mietern, Speicherung von Wohnungsdaten (Etage/Nr., Miete, Nebenkosten), Schlüsselverwaltung.
- **Datenbank:** Speicherung in SQLite mit FOREIGN KEYs für Konsistenz.

1.2. Nicht-funktionale Anforderungen

- **Benutzerfreundlichkeit:** Modernes, intuitives UI mit Angular.
- **Offline-Funktion:** Lokale Speicherung, keine Internetabhängigkeit.
- **Performance:** Schnelle Ladezeiten, effiziente Abfragen mit sqlite3.

1.3. Technische Anforderungen

- **Stack:** Electron (Windows), Angular (Frontend), sqlite3 (DB), IPC für Datenbankzugriff

2. Projektbeschreibung

Im Rahmen dieses Projekts wurde eine Desktop-Anwendung für Gebäudemanager entwickelt, die eine effiziente Verwaltung von Mietobjekten ermöglicht. Die Anwendung nutzt eine lokale SQLite-Datenbank als Datei, sodass alle Daten direkt auf dem Rechner gespeichert werden und keine Internetverbindung erforderlich ist.

Technologisch basiert das Projekt auf Electron, wodurch die Anwendung als plattformübergreifende Desktop-App bereitgestellt werden kann. Das Frontend wurde mit Angular entwickelt, um eine moderne und benutzerfreundliche Oberfläche zu gewährleisten. Die Datenbankanbindung erfolgt über sqlite3, wobei Daten zu Gebäuden, Mietern und Wohnungen strukturiert erfasst und verarbeitet werden.

Durch diese Kombination aus Technologien bietet die Anwendung eine stabile und performante Lösung für Gebäudemanager, die ihre Daten lokal und unabhängig verwalten möchten.

3. Dokumentation des Projekts

Dieses Projekt basiert auf Electron für den Desktop-Teil (Main-Prozess) und Angular für das Frontend (Renderer-Prozess). Die Hauptidee ist, eine Anwendung zu entwickeln, die sowohl offline als auch performant läuft und dabei eine SQLite-Datenbank nutzt.

3.1. Aufbau des Main-Prozesses

- **database/**: Enthält die DatabaseService-Klasse, die den Zugriff auf SQLite regeln. Hier wurde bewusst Logik “ausgelagert”, damit sie global im Main-Prozess verfügbar ist. Ansonsten befindet sich in diesem Ordner noch eine Datei, welche die SQL-Statement-Konstanten beinhaltet, sodass diese übersichtlich in der DatabaseService-Klasse eingefügt werden können. Somit bleibt der Code in der DatabaseService-Klasse leserlicher.
- **event-handler/**: Hier liegt der IPC-Handler, der Befehle aus dem Renderer-Prozess entgegennimmt. Durch diese Auslagerung bleibt der Code in der main.ts leserlicher. Zusätzlich kann man in diesem Ordner weitere Event-Handler einbauen. Somit weiß man, dass diese Handler sich stets in diesem Ordner befinden.
- **main-util/**: Beinhaltet Hilfsfunktionen oder Schnittstellen (*interfaces*), die der Main-Prozess an verschiedenen Stellen benötigt. Auch dies ist eine globale Auslagerung, um Code-Duplikate zu vermeiden.
- **main.ts**: Startpunkt der Electron-App. Hier wird das BrowserWindow erstellt, das Angular-Frontend geladen und die IPC-Handler registriert.

  (s. Anhang: *Abbildung 1)*

3.2. Aufbau des Renderer-Prozesses

- **core/**: Enthält gemeinsam genutzte Ressourcen des Frontends wie *components*, *services*, *interfaces* und *utils*. Diese Struktur sorgt für eine saubere Trennung und einfache Wartbarkeit.
- **portal/**: Beinhaltet die eigentlichen Angular-Komponenten (z. B. app.component.html, app.component.scss usw.), die im UI angezeigt werden. Über *Angular* wird das

Frontend gerendert und durch IPC-Aufrufe mit dem Main-Prozess (Electron) kommuniziert.

(s. Anhang: *Abbildung 2)*

3.3. Gemeinsame Nutzung des *shared*-Ordner

Im **Root-Verzeichnis** gibt es einen Ordner namens shared. Darin befindet sich Code, der sowohl vom Main-Prozess (Backend) als auch vom Renderer-Prozess (Frontend) verwendet wird – zum Beispiel Typ-Definitionen und Interfaces. Dadurch bleiben wichtige Strukturen konsistent, und man vermeidet redundanten Code an mehreren Stellen.

3.4. Datenbank & Kommunikation

- **SQLite** wird lokal eingesetzt, um die Daten zu speichern. In DatabaseService wird dafür gesorgt, dass Tabellen existieren und bei Bedarf erstellt werden (z. B. für Gebäude, Mieter und Wohnungen).
- **IPC (Inter Process Communication)** verbindet Angular (Renderer) mit Electron (Main). Die Methoden, die in event-handler/ registriert werden, rufen im Hintergrund DatabaseService auf und geben das Ergebnis an das Frontend zurück.

  (s. Anhang: *Abbildung 3)*

3.5. Abschließende Bemerkungen

Durch die klare Strukturierung und das Auslagern wiederverwendbarer Teile (z. B. Datenbankzugriff, IPC-Handler, gemeinsame *shared*-Ressourcen) bleibt das Projekt leicht erweiterbar und gut wartbar. Diese Trennung der Verantwortlichkeiten sorgt zudem dafür, dass sich Frontend und Backend unabhängig weiterentwickeln lassen, ohne dass Code mehrfach geschrieben werden muss. Auf diese Weise ist das Projekt übersichtlich organisiert, performant und bereit für zukünftige Erweiterungen.

4. Benutzerhandbuch

4.1. Dashboard

Das Dashboard bildet die zentrale Anlaufstelle der Anwendung und ermöglicht eine intuitive Navigation zu den verschiedenen Unterseiten (s. Anhang: *Abbildung 4*). Die wichtigsten Funktionen sind über deutlich erkennbare Icons zugänglich, die es dem Nutzer erlauben, mit nur einem Klick zur gewünschten Seite zu gelangen. Zusätzlich befindet sich in der oberen Navigationsleiste eine weitere Möglichkeit, zwischen den verschiedenen Bereichen zu

wechseln. Falls sich der Nutzer auf einer Unterseite befindet und zurück zur Hauptansicht gelangen möchte, kann er dies jederzeit tun, indem er auf das Logo oder das Bild oben links im Headerbereich klickt – diese Funktion führt ihn unmittelbar zurück zum Dashboard. Da das Dashboard selbst keine weiterführenden Interaktionen bietet, dient es ausschließlich als Übersichts- und Weiterleitungsseite zu den Kernfunktionen der Anwendung.

4.2. Hauptbereiche

Die Anwendung gliedert sich in die drei Hauptbereiche: Gebäude, Wohnungen und Mieter. Jede dieser Seiten ist nach einem einheitlichen Prinzip gestaltet, um eine konsistente Benutzererfahrung zu gewährleisten. Diese Seiten (s. Anhang: *Abbildung 4*) bieten folgende Funktionen:

4.3. Einträge hinzufügen

Um einen neuen Eintrag hinzuzufügen, öffnen Sie zunächst die entsprechende Unterseite. Klicken Sie anschließend oben links über der Tabelle auf „XYZ hinzufügen“. Daraufhin öffnet sich ein Dialog-Fenster (s. Anhang: *Abbildung 5*) mit einem Formular, in das Sie die erforderlichen Informationen eingeben. Nachdem Sie das Formular abgesendet haben, wird der Eintrag gespeichert. Falls der neue Eintrag nicht sofort sichtbar ist, klicken Sie oben rechts über der Tabelle auf „Neu laden“, um die Anzeige zu aktualisieren.

4.4. Einträge löschen

Zum Löschen eines Eintrags öffnen Sie die entsprechende Unterseite und suchen den gewünschten Datensatz in der Tabelle. Klicken Sie in der Zeile des Eintrags auf das rote Löschen-Icon und bestätigen Sie die Aktion im erscheinenden Dialog-Fenster. Falls der gelöschte Eintrag weiterhin angezeigt wird, können Sie die Ansicht durch Klicken auf „Neu laden“ aktualisieren.

4.5. Aktualisierung der Daten

Falls ein bestehender Eintrag geändert werden soll, öffnen Sie die Unterseite und klicken in der entsprechenden Zeile auf das Stift-Icon zum Bearbeiten. In dem sich öffnenden Dialog-Fenster sind die aktuellen Daten bereits eingetragen, sodass Sie nur die gewünschten Änderungen vornehmen müssen. Nach dem Absenden des

Formulars wird der Eintrag aktualisiert. Falls die Änderungen nicht direkt in der Tabelle sichtbar sind, klicken Sie auf „Neu laden“, um die neuesten Daten anzuzeigen.

5. Dokumentation des Projektablaufs

5.1. Projektstart & Planung

Zu Beginn des Projekts stand die Festlegung der Anforderungen. Die Software sollte eine effiziente Verwaltung von Gebäuden, Mietern und Wohnungen ermöglichen. Nach der Definition der grundlegenden Funktionen, wie das Erstellen, Bearbeiten und Löschen von Datensätzen, wurde die Technologieauswahl getroffen. Für dieses Projekt entschieden wir uns für Electron als Desktop-Framework, Angular für das Frontend und sqlite3 für die lokale Datenbank. Ein grober Projektplan wurde erstellt, in dem die wichtigsten Meilensteine und Arbeitsschritte festgehalten wurden.

5.2. Konzeption & Architektur

Im nächsten Schritt konzentrierten wir uns auf die Datenbankstruktur und die Systemarchitektur. Eine zentrale Aufgabe war der Entwurf der Tabellen für Gebäude, Mieter und Wohnungen, um die Beziehung zwischen den verschiedenen Entitäten zu organisieren. Ein wichtiger Aspekt war auch die Planung der IPC-Kommunikation zwischen dem Frontend und dem Backend. Die Kommunikation wurde so konzipiert, dass Daten sicher und effizient zwischen den Prozessen übertragen werden können. Außerdem wurde ein UI/UX-Design für das Dashboard sowie die Unterseiten entworfen, dass eine klare Navigation und benutzerfreundliche Interaktionen ermöglicht.

5.3. Implementierung

Die Implementierung des Projekts erfolgte in mehreren Phasen, die durch folgende Meilensteine strukturiert wurden:

Meilenstein 1: Erstellung der Datenbank & Backend-Logik

Zu Beginn wurde die SQLite-Datenbank erstellt, wobei Tabellen für die Speicherung von Gebäude-, Mieter- und Wohnungsinformationen definiert wurden. Mit der

Integration des sqlite3-Packages wurde die Datenbankanbindung realisiert. Nach der Implementierung erfolgten erste Tests, um sicherzustellen, dass alle Daten korrekt gespeichert und abgerufen werden können.

Meilenstein 2: Umsetzung des Frontends mit Angular

Parallel zur Backend-Entwicklung begann die Gestaltung des Frontends mit Angular. Die Anwendung erhielt ein Dashboard, das als zentrale Navigationsplattform dient. Zudem wurden Seiten für die Verwaltung von Gebäuden, Wohnungen und Mietern erstellt. Formulare zur Dateneingabe und Tabellenansichten zur Darstellung der gespeicherten Daten wurden ebenfalls implementiert.

Meilenstein 3: Integration von Backend & Frontend

Nach der Entwicklung beider Teile der Anwendung begann die Integration von Backend und Frontend. Das Electron-Framework ermöglichte die Kommunikation zwischen den beiden Prozessen. Dadurch konnten die Benutzereingaben im Frontend direkt in die Datenbank geschrieben und von dort abgerufen werden.

5.4. Qualitätssicherung & Optimierung

Nach den ersten Tests und der Behebung identifizierter Fehler wurde der Fokus auf Qualitätssicherungsmaßnahmen gelegt, um die Stabilität und Benutzerfreundlichkeit der Anwendung zu gewährleisten.

Meilenstein 4: Verbesserung der Stabilität & Benutzerfreundlichkeit

Im Rahmen der Qualitätssicherung wurden regelmäßig eigenständige Code-Reviews durchgeführt, um die Qualität des Codes zu sichern und mögliche Fehler frühzeitig zu erkennen. Zudem wurden Logging-Mechanismen implementiert, um Fehler und Probleme schnell identifizieren zu können. Die Benutzeroberfläche wurde basierend auf Nutzerfeedback weiter optimiert, um eine intuitive und benutzerfreundliche Bedienung zu gewährleisten.

5.5. Abschluss & Dokumentation

Der Abschluss des Projekts bestand in der Dokumentation und der Übergabe der fertigen Anwendung.

Meilenstein 5: Fertigstellung & Bereitstellung der Software

Zum Abschluss des Projekts wurde ein ausführliches Benutzerhandbuch erstellt, das den Endnutzern alle wichtigen Funktionen der Software näherbringt. Eine detaillierte Code- und Architektur-Dokumentation wurde angefertigt, um die Wartung und zukünftige Erweiterungen der Software zu erleichtern. Abschließend wurde die Anwendung bereitgestellt und das Projekt präsentiert.

Dieser strukturierte Ablauf mit klar definierten Meilensteinen hat sichergestellt, dass das Projekt effizient und innerhalb des Zeitrahmens erfolgreich umgesetzt werden konnte

6. Qualitätssicherungsmaßnahmen

Aufgrund der zeitlichen Differenz, die durch die fehlerhafte Anbindung der Datenbank resultierte, musste ich leider zeitliche Abzüge bei der Qualitätssicherung des Projektes machen. Zwar war es nicht möglich, ein umfassendes Testverfahren oder eine intensive Qualitätssicherung durchzuführen, dennoch hat man Wert daraufgelegt, potenziellen Problemen in der Datenbankstruktur vorzubeugen. So wird beim Start der Anwendung stets überprüft, ob alle benötigten Tabellen vorhanden sind; sollten sie fehlen, werden sie automatisch erstellt. Auf diese Weise wird sichergestellt, dass dem Kunden jederzeit eine funktionsfähige Datenbasis zur Verfügung steht und mögliche Fehler in diesem Bereich keine Blockade verursachen. Darüber hinaus wurden gemeinsame Ressourcen (z. B. Code im *shared*-Ordner) genutzt, damit Änderungen an einer Stelle automatisch an allen relevanten Stellen greifen und Probleme frühzeitig erkannt werden. Dies stellt eine wichtige Grundlage für Stabilität und Zuverlässigkeit dar, auch wenn weiterführende Qualitätstests aus Zeitgründen nur in begrenztem Umfang stattfinden konnten.

7. Reflexion

Im Rahmen dieses Projekts habe ich eine Datenbank entworfen, ein benutzerfreundliches Frontend mit Angular entwickelt und versucht, die Datenbank über ein IPC-Konzept mithilfe des Electron-Frameworks in die Anwendung zu integrieren. Dabei lag der Fokus nicht nur auf

der Konzeption und Umsetzung der Datenbank selbst, sondern auch auf der Anbindung über IPC (inter-process communication). Es gab sowohl positive als auch herausfordernde Aspekte, die ich im Folgenden reflektiere.

7.1. Positive Erkenntnisse

Ein großer Erfolg war die Konzeptionierung der Datenbank. Bereits in der Planungsphase konnte ich eine klare Struktur entwickeln, die sowohl eine effiziente Speicherung als auch eine schnelle Verarbeitung der Daten ermöglicht. Die logische Modellierung und Normalisierung der Datenbank haben sich als sinnvoll erwiesen, da sie für eine gute Performance und Wartbarkeit sorgen. Die Tabellen für Gebäude, Mieter und Wohnungen wurden mit einer klaren Struktur erstellt, wobei wichtige Beziehungen durch FOREIGN KEY-Constraints sichergestellt wurden. Dadurch wurde eine solide Datenbasis geschaffen, die eine effiziente Verwaltung der Wohnobjekte ermöglicht. Die saubere Normalisierung sorgt für gute Wartbarkeit und Skalierbarkeit der Datenbank.

Auch das Design des Frontends verlief ohne größere Probleme. Da ich bereits Erfahrung mit Angular hatte, konnte ich das Framework effizient einsetzen, um eine benutzerfreundliche Oberfläche zu gestalten. Dadurch konnte ich mich mehr auf die Funktionalität und die Nutzerführung konzentrieren, anstatt mich erst in neue Technologien einzuarbeiten. Dabei habe ich darauf geachtet, dass das Design nicht nur optisch ansprechend ist, sondern auch eine intuitive Nutzung ermöglicht. Die Nutzerführung und das Layout wurden so konzipiert, dass sich User schnell zurechtfinden können.

7.2. Herausforderungen

Eine große Herausforderung war die Einarbeitung in das IPC-Konzept im Zusammenhang mit Electron. Da das Thema technisch anspruchsvoll ist und eine tiefere Auseinandersetzung mit Inter-Prozess-Kommunikation erfordert, fiel es mir schwer, direkt einen klaren Zugang dazu zu finden. Die theoretischen Grundlagen mussten zunächst verstanden und anschließend in den praktischen Kontext meines Projekts übertragen werden. Die Kommunikation zwischen Renderer- und Main-Prozess war anfangs nicht ganz intuitiv, insbesondere in Kombination mit dem Zugriff auf die SQLite-Datenbank. Da ich zuvor gar keine Erfahrung mit Electron machen durfte, musste ich mich erst in die Inter-Prozess-Kommunikation einarbeiten, die ich für die Entwicklung meiner Applikation benötigte.

Auch die Anbindung der Datenbank im IPC-Main mit sqlite3 (Node-Package) bereitete mir viele Schwierigkeiten. Insbesondere die Pfadverwaltung war problematisch, da der Datenbankzugriff nicht sofort reibungslos funktionierte. Es erforderte Debugging-Sessions über mehrere Tage hinweg und Anpassungen, um sicherzustellen, dass die Anwendung die

Datenbank korrekt findet und darauf zugreifen kann. Dies erwies sich als zeitintensiv, da nicht immer sofort ersichtlich war, wo genau Probleme auftraten. Schließlich beanspruchte dies die meiste Zeit im Projekt.

7.3. Fazit

Trotz der Herausforderungen war das Projekt insgesamt eine wertvolle Lernerfahrung. Die erfolgreiche Umsetzung der Datenbankstruktur und des Frontends zeigt, dass eine solide Basis geschaffen wurde. Die Schwierigkeiten mit Electron und sqlite3 haben mir zudem einen tieferen Einblick in die Herausforderungen der Inter-Prozess-Kommunikation und des Datenbankzugriffs in einer Desktop-Anwendung gegeben. Mit mehr Erfahrung und Vorbereitung in diesen Bereichen hätte ich einige Probleme möglicherweise schneller lösen können. Dennoch hat das Projekt mein technisches Verständnis erweitert und mir wertvolle Erkenntnisse für zukünftige Projekte geliefert.

Anfangs hatte ich das Gefühl, dass das Thema zu groß und komplex für mich sein könnte und dass ich möglicherweise keinen Anfang setzen könnte. Doch indem ich drangeblieben bin, konnte ich stets mehr Erkenntnisse gewinnen, dass alles erlernbar ist, wenn man sich intensiv damit auseinandersetzt. Diese Erfahrung werde ich für zukünftige Projekte mitnehmen, um mich nicht von anfänglichen Schwierigkeiten entmutigen zu lassen.

Ich habe auch erkannt, dass ich lernen muss, mich selbst zu motivieren, wenn ich vor schwierigen Aufgaben stehe und Dinge zunächst unlösbar erscheinen. Besonders wichtig wird das in Zukunft sein, wenn ich in einem Unternehmen neue Projekte mit unbekannten Technologien übernehme – vor allem dann, wenn sonst niemand Erfahrung damit hat und ich keine direkte Bezugsperson wie Herrn Clausen an meiner Seite habe. Ich möchte daran arbeiten, mehr Vertrauen in meine eigene Lernfähigkeit zu entwickeln und mich nicht von Unsicherheiten ausbremsen zu lassen.

8. Anhang

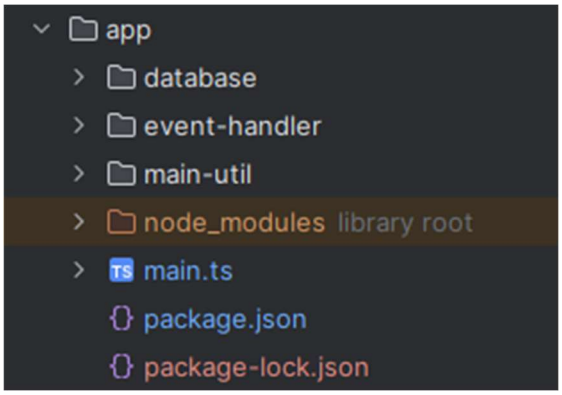

Abbildung 1: Ordnerstruktur des Main-Prozesses

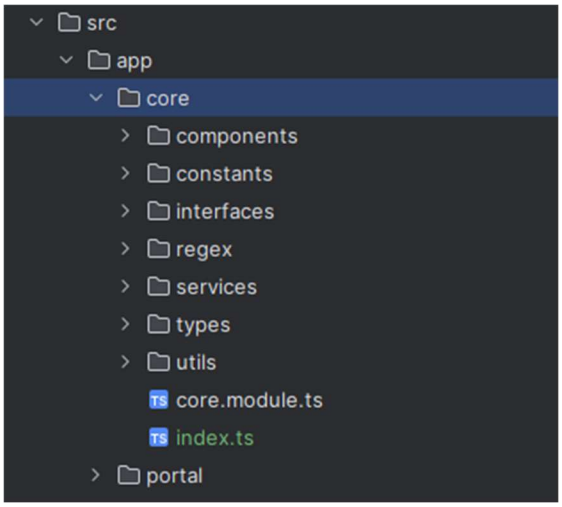

Abbildung 2: Ordnerstruktur des Renderer-Prozesses (nicht vollständige Ansicht)

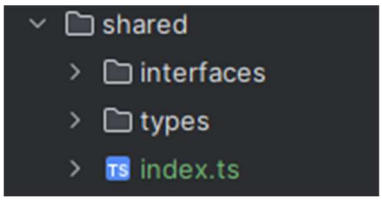

Abbildung 3: Ordnerstruktur des shared-Ordners

Abbildung 4: Unterseite (Bsp. Gebäude-Seite)

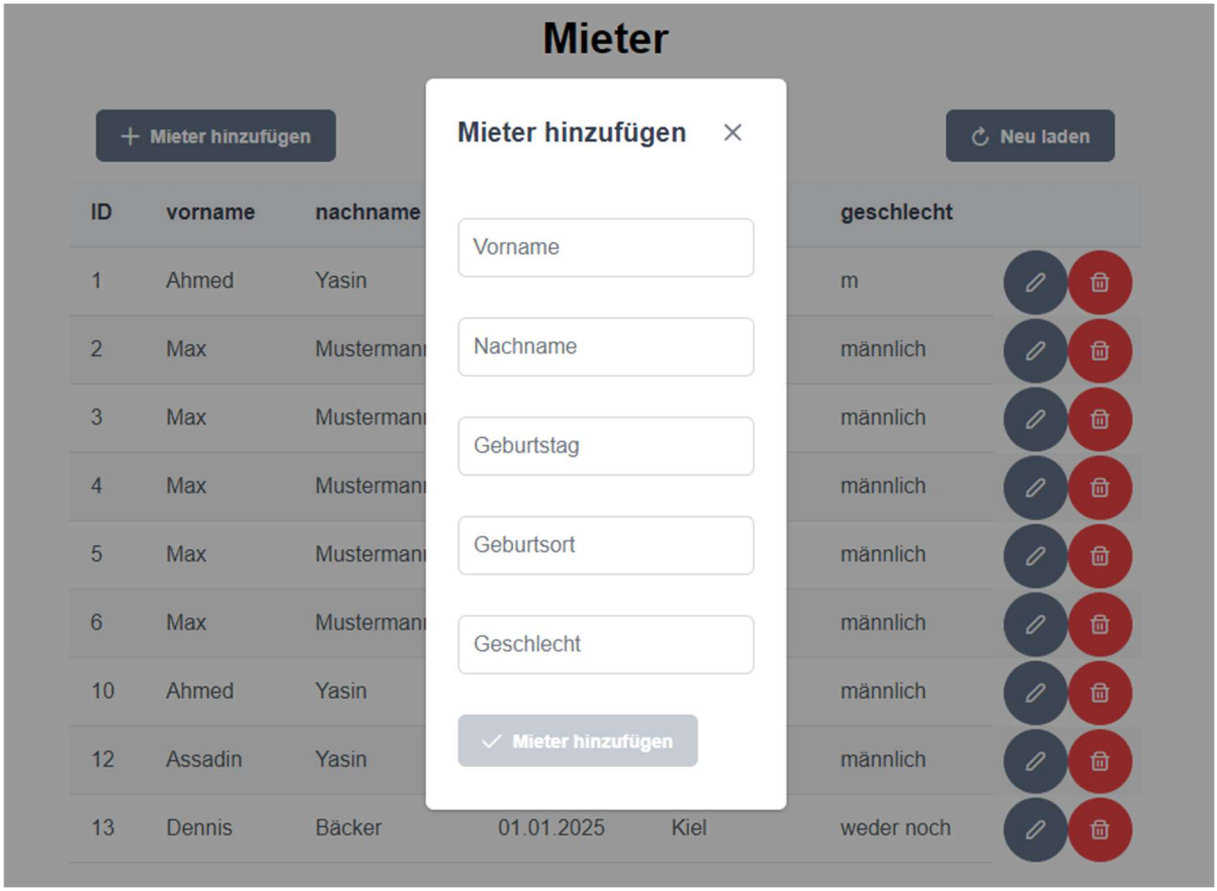

Abbildung 5: Dialog-Fenster zum Erstellen oder Bearbeiten eines Eintrags