Informatik
Hauptcampus

HOCHSCHULE TRIER

Nutzerzentrierte Entwicklung eines offline-fähigen Softwaresystems zur Erstellung von technischen Gebäudebestandsaufnahmen

User-centered development of an offline-capable software system for creating technical building inspections

Jens Becker

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr. rer. nat. Tilo Mentler

Trier, 05.08.2024

Kurzfassung

Die Erstellung von technischen Gebäudebestandsaufnahmen im Rahmen von Sanierungsfahrplänen bei der Ingenieurgesellschaft TRAGWERK in Trier erfolgt derzeit handschriftlich auf mehrseitigen PDF-Checklisten. Die handschriftliche Erfassung bringt jedoch den Nachteil mit sich, dass die Notizen in einer Nachbereitungsphase im Büro in eine digitale Form überführt werden, was zeitaufwändig ist. Ebenso führen die handschriftlich ausgefüllten Checklisten teilweise zu ungenauen und unvollständigen Bestandsaufnahmen und erschweren die Zusammenarbeit mehrerer Mitarbeiter an einem Projekt. Das Ziel dieser Bachelorarbeit ist die Entwicklung eines offline-fähigen Softwaresystems zur digitalen Erstellung von technischen Gebäudebestandsaufnahmen. Diese Lösung soll die beschriebenen Probleme eliminieren, die Nutzungsanforderungen erfüllen und nach Abschluss der Arbeit im Arbeitsalltag des Praxispartners verwendet werden können. Um dieses Ziel zu erreichen, wird ein nutzerzentriertes, iteratives Entwicklungsvorgehen angewendet. Durch Feldstudien zur Analyse des Arbeitsablaufs und der Durchführung einer Nutzungskontextanalyse werden Nutzungsanforderungen an die Software abgeleitet. Ein Konzept in Form von Wireframes unterstützt die Implementierung, die schrittweise durch formative Nutzungstests evaluiert und verbessert wird. Die Nutzungskontextanalyse ergab, dass die Mitarbeiter und zukünftigen Nutzer der Software in zwei rollenbasierte Benutzergruppen eingeteilt werden können: Datenerfasser, die vor Ort offline mit einem iPad arbeiten, und Projektbearbeiter, die im Büro am PC den Sanierungsfahrplan erstellen. Die Evaluation durch Nutzungstests zeigte, dass die entwickelte Web- und iOS-App die Effizienz des Prozesses durch strukturierte digitale Formulare mit Handschrifterkennung, Auswahlfeldern, duplizierbaren Formulargruppen und bedingten Feldern erhöht. Dies verbessert die Vollständigkeit und Genauigkeit der Datenerfassung, beschleunigt die Projektbearbeitung im Büro und erleichtert die Zusammenarbeit mehrerer Mitarbeiter an einem Projekt.

Abstract

The creation of technical building inspections at the engineering company TRAGWERK in Trier is currently done by handwriting on multipage PDF checklists. The disadvantage of handwritten recording is that the notes have to be converted into digital form in a follow-up phase in the office, which is time-consuming. Likewise, handwritten checklists sometimes lead to inaccurate and incomplete inspections and complicate project collaboration. The aim of this bachelor thesis is to develop an offline-capable software system for the digital creation of building inspections. This solution should eliminate the described problems, meet user requirements, and be usable in the day-to-day work of the practice partner after completion. A user-centered, iterative development approach is used to achieve the goal. Field studies to analyze the workflow and a context of use analysis lead to deriving user requirements for the software. A concept in the form of wireframes supports the implementation, which is gradually evaluated and improved through formative user tests. The analysis revealed that employees can be divided into two role-based user groups: Data collectors, who work offline on site with an iPad, and project workers, who create the renovation roadmap on a PC in the office. The evaluation showed that the developed web- and iOS-app increases the process efficiency through structured digital forms with handwriting recognition, selection fields, duplicable form groups, and conditional fields. This improves the completeness and accuracy of data collection, speeds up project processing in the office, and promotes collaboration.

Inhaltsverzeichnis

1

Einleitung

1.1 Motivation

In einer zunehmend digitalisierten Welt stehen Unternehmen vor der Herausforderung, ihre Arbeitsprozesse effizienter zu gestalten, um wettbewerbsfähig zu bleiben. Dabei können Softwaresysteme zum Einsatz kommen, deren Nutzung nicht nur erhebliche Zeit- und Kostenersparnisse mit sich bringen kann, sondern auch das Potenzial hat, die Qualität und Genauigkeit der Arbeitsprozesse zu verbessern. Diese Software sollte nicht nur technisch einwandfrei funktionieren, sondern auch im Praxiseinsatz praktikabel und tauglich sein.
Einer der Hauptgründe für den Misserfolg von Systemen ist ein unangemessenes oder unvollständiges Verständnis der Benutzererfordernisse. [ISO20]
Dieser Thematik widmet sich diese Arbeit, indem ein offline-fähiges Softwaresystem zur Erstellung von technischen Gebäudebestandsaufnahmen nach einem nutzerzentrierten Entwicklungsprozess entwickelt wird. Dabei werden die Anforderungen, Umgebung und Bedürfnisse der Nutzer in allen Phasen der Entwicklung berücksichtigt.

1.2 Problemstellung

Die technische Gebäudebestandsaufnahme ist ein Prozess in der Bau- und Ingenieurbranche, der dazu dient, den aktuellen Zustand eines Gebäudes vor Ort zu erfassen und zu dokumentieren. Dieser Prozess ist etwa bei energetischen Sanierungen relevant, da die erhobenen Daten die Grundlage für die Planung und Durchführung von Maßnahmen zur Verbesserung der Energieeffizienz darstellen.
Die Bestandsaufnahme umfasst die Erfassung verschiedener Datenpunkte zu den jeweiligen Bauteilen eines Gebäudes, wie beispielsweise energetische Eigenschaften und technische Anlagen.
Bei der Ingenieurgesellschaft TRAGWERK in Trier, welche der Praxispartner dieser Arbeit ist, wird die technische Gebäudebestandsaufnahme von Wohngebäuden derzeit mithilfe von mehrseitigen PDF-Checklisten durchgeführt. Während der Vor-Ort-Termine bei Kunden werden die relevanten Daten über die Bauteile des Hauses handschriftlich in Formularen notiert. Diese Formulare enthalten

Kategorien mit Feldern, die systematisch abgearbeitet werden, um ein umfassendes Bild des Gebäudes zu erhalten. Nach Abschluss der Vor-Ort-Termine werden die erfassten Daten im Büro zunächst abgetippt und im späteren Verlauf in eine Energieberater-Software übertragen, um sie für weitere Analysen und Planungen zu nutzen.
Die Methode der handschriftlichen Datenerfassung vor Ort bringt jedoch mehrere Nachteile und Probleme mit sich, die sich negativ auf die Effektivität und Effizienz des Arbeitsablaufs und der Qualität der erfassten Daten auswirken. Zum einen müssen die handschriftlichen Notizen nach einem Vor-Ort-Termin in einer Nachbereitungsphase im Büro in eine digitale Form überführt werden, was zeitaufwändig ist. Darüber hinaus führen die handschriftlich ausgefüllten Checklisten teilweise zu ungenauen und unvollständigen Bestandsaufnahmen und erschweren zudem die Zusammenarbeit von mehreren Mitarbeitern an einem Projekt.

1.3 Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, ein offline-fähiges Softwaresystem mit der Hauptfunktionalität der Vor-Ort Gebäudebestandsaufnahme zu entwickeln, das die genannten Probleme eliminiert, die Nutzungsanforderungen erfüllt, technisch einwandfrei funktioniert und nach Abschluss dieser Arbeit beim Praxispartner im Arbeitsalltag verwendet werden kann.
Das System soll die Genauigkeit, Vollständigkeit und Effizienz der Datenerfassung steigern und sich in den bestehenden Arbeitsablauf der Nutzer integrieren und diesen erweitern.
Im Rahmen dieser Arbeit soll herausgefunden werden, wie ein solches System gestaltet und entwickelt werden muss, um eine möglichst hohe Benutzerakzeptanz und Tauglichkeit im Praxiseinsatz zu erreichen.

1.4 Vorgehen

Um das zuvor genannte Ziel zu erreichen, wird ein nutzerzentriertes, iteratives Entwicklungsvorgehen angewendet, bei dem schrittweise analysiert, entwickelt, evaluiert und verbessert wird.
Ein besonderer Fokus liegt zunächst auf der Untersuchung des Nutzungskontextes durch das Verständnis und der Analyse des aktuellen Arbeitsablaufs. Dazu werden die Mitarbeiter der Ingenieurgesellschaft TRAGWERK in ihrem Arbeitsablauf begleitet und befragt. Auf Basis dieser Analyse werden Nutzungsanforderungen an das zu entwickelnde System abgeleitet.
Basierend auf den Erkenntnissen der Analyse wird ein Gestaltungskonzept entwickelt und in mehreren Phasen technisch implementiert, wobei nach Abschluss jeder Implementierungsphase ein Prototyp des Systems im Praxiseinsatz mithilfe von Feldtests formativ evaluiert wird. Die dabei gewonnenen Erkenntnisse fließen in die weitere Entwicklung ein, um sicherzustellen, dass das System im Praxisalltag tauglich ist.

2

Stand der Forschung und Entwicklung

2.1 Verwandte Wissenschaftliche Arbeiten

2.1.1 Bauwerksinformationsmodellierung (BIM)

Die Erfassung und Speicherung technischer Gebäudedaten ist mit dem Konzept des „Building Information Modeling“, zu Deutsch Bauwerksinformationsmodellierung, abgekürzt „BIM“ verwandt.
BIM ist eine Methode zur digitalen Erstellung und Verwaltung von Bauwerksdaten mithilfe spezialisierter Softwarewerkzeuge. Dabei wird ein umfassendes digitales Abbild eines Bauwerks erstellt, welches typischerweise die dreidimensionale Geometrie der Bestandteile des Bauwerks enthält. Neben der 3D-Geometrie werden die enthaltenen Bauteile ebenso mit weiterführenden Informationen zur Bedeutung (Semantik) oder Ausprägung versehen. [BKKB21]
Im Jahr 2015 empfahl die Reformkommission Bau von Großprojekten, beauftragt von der Bundesregierung, in ihrem Abschlussbericht die Nutzung von BIM, um zukünftige Großprojekte innerhalb des geplanten Zeit- und Kostenrahmens abzuwickeln. [BKKB21]
Die Arbeit „BIM im Gebäudebestand - Herausforderungen in der Sanierung“ aus dem Jahr 2023 zeigt, dass die BIM-Methode auch die Effizienz bei Sanierungsaufgaben steigern kann, wobei ein wesentlicher Vorteil die Wiederverwertbarkeit von strukturierten Daten in BIM-Modellen ist. Allerdings wird auch auf fehlende Informationen über Bestandssanierung in den BIM-spezifischen Normen und die mangelnden öffentlichen Erfahrungswerte zur Anwendung der BIM-Methode im Bestandsbau hingewiesen. [CRS]

Fazit

Während BIM zunehmend in die Baubranche Einzug hält und hauptsächlich auf komplexen 3D-Modellen basiert, die am PC erstellt und bearbeitet werden, konzentriert sich die vorliegende Arbeit auf die textuelle Erfassung technischer Gebäudebestandsdaten von Wohnhäusern zur energetischen Sanierung mithilfe einer mobilen Anwendung vor Ort. Der Fokus liegt hierbei auf der effizienten Datenerfassung, die später im Büro in BIM-kompatible Software übertragen werden kann. Durch die Entwicklung dieses Softwaresystems wird eine Brücke zwischen der vor

Ort durchgeführten Bestandsaufnahme und der komplexen BIM-gestützten Datenverwaltung im Büro geschlagen.

2.1.2 Web-basierte Bestandsaufnahme App

Ein weiterer relevanter Beitrag zur Thematik der Erfassung und Speicherung technischer Gebäudedaten stammt von Pereira et al. (2022). Dieser Beitrag beschreibt die Entwicklung einer mobilen Anwendung zur Erfassung von Mängeln in der Gebäudehülle durch Gutachter am Inspektionsort. Obwohl sich diese Bachelorarbeit nicht auf mit der Erfassung von Mängeln in der Gebäudehülle befasst, zeigt der Artikel dennoch parallelen mit dieser Arbeit und ist daher erwähnenswert.
Zunächst wird thematisiert, dass die Verwendung eines Inspektionssystems, das auf Papierformularen basiert, einige Schwierigkeiten mit sich bringt. Diese umfassen die Menge an mitzuführenden Dokumenten, die hohe Fehleranfälligkeit sowie die Notwendigkeit, die Daten später in ein Computersystem zu übertragen.
Zur Lösung dieser Probleme wurde eine webbasierte Softwareanwendung entwickelt, die ein strukturiertes Inspektionssystem für die Erfassung von Mängeln an der Gebäudehülle bietet. Das Inspektionsformular ist dabei in drei Hauptteile gegliedert: die Charakterisierung des Gebäudes, des Gebäudeelements und dessen Zustandsverschlechterung.
Tests dieser Anwendung zeigten mehrere Vorteile: Die Erfassung der Mängel über die Web-Anwendung spart Zeit, da die Notwendigkeit der späteren Dateneingabe entfällt. Außerdem wird der Gutachter durch die Struktur der Anwendung durch die Prüfung geführt. Zuletzt läuft diese auf erschwinglichen Geräten und ermöglicht die direkte Speicherung von Informationen, wodurch Fehler und Auslassungen bei den Inspektionsdaten reduziert werden, da das System deren Konsistenz garantiert. [PSS+22]

Fazit

Die Arbeit von Pereira et al. (2022) verdeutlicht die Vorteile der digitalen Erfassung von Gebäudedaten vor Ort und zeigt, wie durch den Einsatz einer webbasierten Anwendung Effizienz und Genauigkeit gesteigert werden können.
Die vorliegende Arbeit baut auf den zuvor genannten Erkenntnissen auf, geht jedoch einen Schritt weiter, indem eine offline-fähige App entwickelt wird. Während die Lösung von Pereira et al. webbasiert ist und eine Internetverbindung erfordert, zielt diese Arbeit darauf ab, die Datenerfassung auch in Umgebungen ohne Internetzugang zu ermöglichen. Dies stellt sicher, dass die Datenerfassung auch in Gebäuden und Gebäudeteilen durchgeführt werden kann, an denen keine Internetverbindung besteht, was bei den Bestandsaufnahmen des Praxispartners häufig der Fall ist.

2.1.3 „Local-first software“

Ein entscheidender Aspekt der vorliegenden Arbeit ist die Offline-Fähigkeit der zu entwickelnden Tablet-App. Während heutzutage viele bekannte Cloud-Anwendungen

ihre Daten ausschließlich auf einem Server speichern und damit die Zusammenarbeit, auch in Echtzeit, ermöglichen, bringt dies den Nachteil mit sich, dass die Datenbesitzverhältnisse beim Server und nicht beim Nutzer liegen. [KWvM19]
Vor dem Aufkommen von Cloud- und Web-Apps lief traditionelle Software lokal auf dem Gerät des Nutzers und speicherte sowie las alle Daten lokal, was dem Nutzer vollen Besitz und Kontrolle über seine Daten gewährte. [KWvM19]
Im Artikel „Local-first software: you own your data, in spite of the cloud“ schlägt Martin Kleppmann den Begriff „local first“ für Software vor, die eine Kopie der Daten bei jedem Nutzer auf dem Gerät speichert, wodurch dieser jederzeit und unabhängig vom Netzwerk Daten lesen und schreiben kann. Eine Netzwerkverbindung wird erst bei der späteren Synchronisation der Daten erforderlich. [KWvM19]
Im Rahmen der Begriffsdefinition werden sieben Ideale beschrieben, die bei der Entwicklung von local-first Software angestrebt werden sollten. Zu diesen idealen gehören (1) Keine Ladeanzeigen, (2) Die Arbeit ist nicht auf ein Gerät beschränkt, (3) Das Netzwerk ist optional, (4) Nahtlose Zusammenarbeit mit Kollegen, (5) Zugänglichkeit der Daten auf unbestimmte Zeit, (6) Sicherheit und Datenschutz standardmäßig und (7) der Nutzer behält das ultimative Eigentum und die Kontrolle über die Daten. [KWvM19]
Kleppmann beschreibt Technologien für mobile Apps mit sogenannten Thick Clients, also Apps mit einem Persistenz-Layer wie SQLite, die komplett offlinefähig sind. Er beobachtet, dass viele App-Entwicklungsteams ihre eigenen, oft unzuverlässigen Algorithmen zur Differenzierung, Zusammenführung und Konfliktlösung entwickeln. Ein spezialisierteres Speichersystem kann hier Abhilfe schaffen. [KWvM19]
Beispiele für solche Technologien sind Firebase, CloudKit, Realm und CouchDB. Die Erkenntnisse zeigen jedoch, dass diese Tools entweder den Nutzern nur wenig Kontrolle über ihre Daten geben oder sich nicht weit durchgesetzt haben. Kleppmann und sein Team haben festgestellt, dass keines der bestehenden Datenspeichersysteme für die Anwendungsentwicklung die Local-First-Ideale vollständig erfüllt. Daher hat sein Labor nach einer Lösung gesucht, die alle local-first Ideale erfüllt. [KWvM19]
Ein Vorschlag von Kleppmann ist die Nutzung von Conflict-free Replicated Data Types (CRDTs) als fundamentale Technologie. CRDTs sind Datenstrukturen, die von Grund auf für die Mehrbenutzerumgebung entwickelt wurden. Das Besondere an ihnen ist, dass sie von mehreren Benutzern gleichzeitig genutzt werden können, ohne dass es zu Konflikten kommt. Das CRDT verfolgt alle vorgenommenen Änderungen und synchronisiert die Änderungen mit anderen Geräten im Hintergrund, wenn eine Netzwerkverbindung verfügbar ist. Wenn der Status gleichzeitig auf verschiedenen Geräten geändert wurde, führt das CRDT diese Änderungen automatisch zusammen. [KWvM19]

Fazit

Die Anwendung des Local-First-Prinzips ist für die vorliegende Arbeit von großer Bedeutung, da die Offline-Fähigkeit der App eine zentrale Anforderung darstellt.

Bei der Entwicklung der App werden deshalb die zuvor genannten Ideale angestrebt.

2.2 Vorhandene Softwareprodukte

Die Suche nach Softwarelösungen für die technische Gebäudebestandsaufnahme zeigt, dass ein Produkt, wie es im Rahmen dieser Bachelorarbeit entwickelt wird, im freien öffentlichen Markt nicht existiert. Es ist davon auszugehen, dass andere Firmen in der Baubranche eigene Softwarelösungen für die Datenerfassung im Bestand nutzen, diese sind allerdings nicht öffentlich zugänglich.
Im Bereich der öffentlich benutz- oder erwerbbaren Software gibt es allerdings eine Vielzahl von Anwendungen für das allgemeine Baumanagement und auch Apps für Generalunternehmer. Viele dieser Apps bieten Funktionen wie Baustellentagebücher, Dokumentation der Baustellenfortschritte, Aufgabenverwaltung und Fotodokumentation, aber keine spezifischen Funktionen für die strukturierte Erfassung von Bestandsgebäudedaten.
Dennoch wurden verwandte Softwareprodukte gefunden, von denen zwei im Folgenden näher betrachtet werden.

2.2.1 „Building Inspector“ - App für Objektbegehungen

Mit der kostenpflichtigen App namens „Building Inspector“ der Firma „Chapps“ aus Belgien können bei verschiedenen Arten von Objektbegehungen Daten mithilfe von Checklisten erfasst werden. [Cha]
Da diese App unter den verwandten Softwareprodukten den ähnlichsten Zweck und Nutzungskontext wie jener der zu entwickelnden App aufweist, wird sie im Folgenden kurz beschrieben. Anschließend wird ein Bezug zur im Rahmen dieser Arbeit zu entwickelnden App hergestellt und ein Vergleich gezogen.

Beschreibung verwandter Funktionen

Eine verwandte Funktion, die die App bietet, ist die Möglichkeit, Objekte bzw. Gebäude zu erstellen und zu verwalten. Dabei werden der Gebäudetyp und die Adresse erfasst. Nach der Erstellung können Objektbegehungen geplant und erstellt werden, für welche eine vordefinierte Checkliste verwendet wird. Diese Checklisten können, wie in Abbildung 2.1 zu sehen, über die Web-App erstellt und bearbeitet werden. Diese bestehen aus mehreren Abschnitten, die jeweils sogenannte Kontrollpunkte enthalten.
Die Einstellungsmöglichkeiten eines Kontrollpunktes bestehen aus einem Namen, einem Werttyp und der Angabe, ob dieser Punkt verpflichtend ausgefüllt werden muss. Der Werttyp, also welche Art von Daten erfasst werden sollen, umfasst dabei unter anderem Ja/Nein-Angaben, Datum, Zustand oder die Angabe der Funktionsfähigkeit mit vordefinierten Feldern.
Über die mobile App kann dann eine Begehung unter Verwendung der Checkliste durchgeführt werden. Die Kontrollpunkte werden dabei, wie in Abbildung 2.2

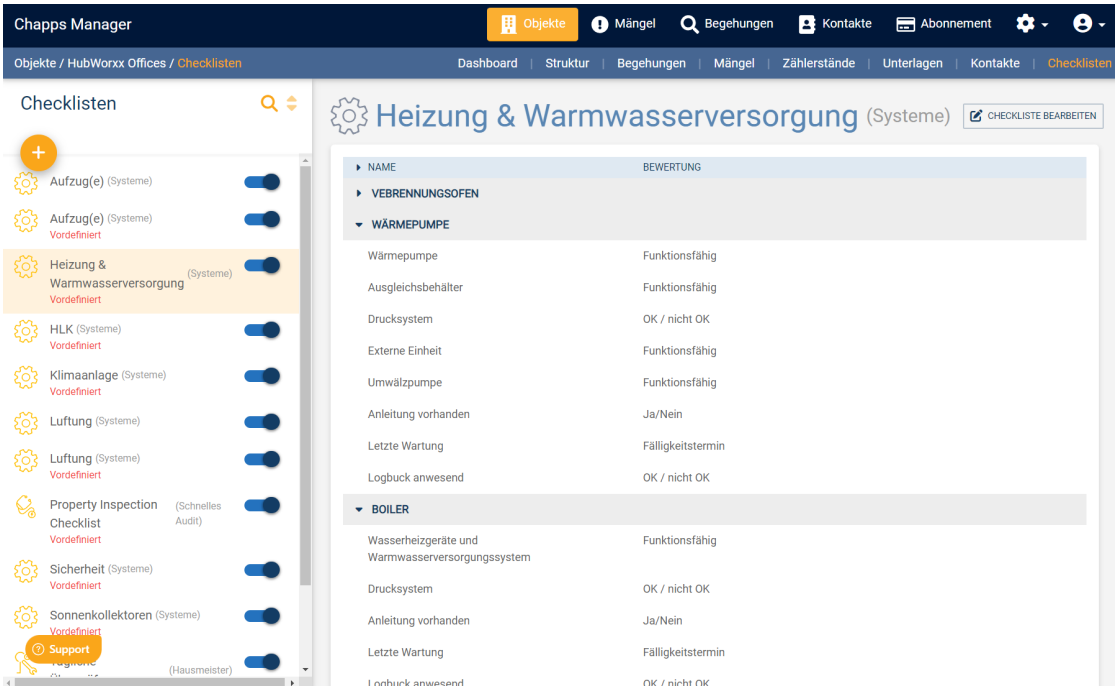


Abbildung 2.1: Verwaltung der Checklisten in der Chapps Web-App

zu sehen, nach Abschnitten gruppiert untereinander angezeigt. Zu jedem Eintrag kann eine Bemerkung, ein Mangel oder ein Bild hinzugefügt werden. Nach Abschluss der Begehung wird aus den Einträgen der Checkliste ein PDF-Protokoll erstellt. Die erfassten Daten werden anschließend synchronisiert und sind über die Web-App einsehbar.

Fazit und Abgrenzung

Die Funktionen der App von Chapps zeigen Parallelen zu den Anforderungen des zu entwickelnden Softwaresystems auf. Die App dient zur Erfassung von Gebäudedaten während einer Begehung vor Ort und ist auch offline funktionsfähig. Des Weiteren wird die Datenerfassung in mehrere Abschnitte unterteilt und es gibt die Möglichkeit zur Erfassung verschiedener Werttypen/Datentypen, die auch in der zu entwickelnden App berücksichtigt werden sollen.
Jedoch gibt es auch wesentliche Unterschiede. Die betrachtete App erfasst Daten in Form von Checklisten und konzentriert sich hauptsächlich auf die Inspektion und Zustandsprüfung von Gebäudeteilen, während die zu entwickelnde App auch technische Daten wie Maße oder Eigenschaften von Gebäuden detailliert erfassen soll. Die App bietet keine Möglichkeit, spezifische Daten wie Größe oder Maßeinheiten als Datentypen für die Kontrollpunkte anzugeben, und erlaubt keine eigene Erstellung von Auswahlmöglichkeiten wie für Angaben zu Baumaterialien oder Konstruktionsarten.
Insgesamt zeigt der Vergleich, dass die App der Firma Chapps eine grundlegende Basis für die Datenerfassung bei Gebäudebegehungen bietet, jedoch in einigen

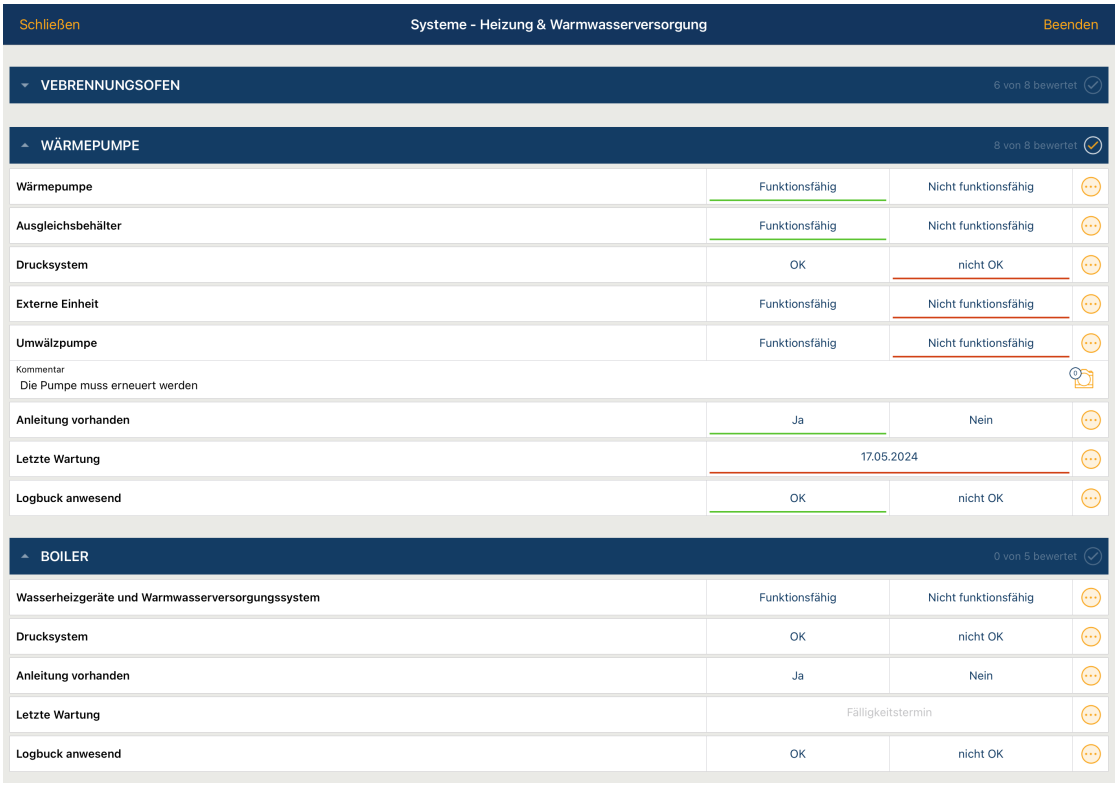


Abbildung 2.2: Datenerfassung bei einer Begehung in der Chapps iOS-App

technischen und anpassbaren Aspekten hinter den Anforderungen der zu entwickelnden App zurückbleibt. Dies unterstreicht die Notwendigkeit einer spezialisierten Lösung, die auf die spezifischen Bedürfnisse des Praxispartners bei der technischen Gebäudebestandsaufnahme zugeschnitten ist.

2.2.2 „FastField“ - Digitale Formulare

Die Software namens „FastField“ der Firma „FastField Inc.“ aus den USA ist eine Softwarelösung für Unternehmen zur Einrichtung und Bereitstellung von Inspektionen, Checklisten und Geschäftsformularen mittels einer Web- und mobilen App. [Fas]
Da diese App unter den verwandten Softwareprodukten die meisten Gemeinsamkeiten mit den Funktionen der zu entwickelnden Software aufweist, wird sie im Folgenden kurz beschrieben. Anschließend wird ein Fazit gezogen und die Abgrenzung zur im Rahmen dieser Arbeit zu entwickelnden Software erläutert.

Beschreibung verwandter Funktionen

Grundsätzlich lassen sich mit der Software Formulare per Web-App erstellen und über die mobile App ausfüllen. Bei der Erstellung können Formulare in Seiten und Abschnitte eingeteilt werden. Jeder Abschnitt kann aus beliebig vielen frei wählbaren Formularfeldern bestehen. Zu den Formularfeldern gehören unter anderem

Text-, Nummern-, Auswahl- und Fotofelder. Die einzelnen Felder können konfiguriert werden, beispielsweise ob sie verpflichtend ausgefüllt werden müssen oder einen Standardwert haben. Die Erstellung eines Formulars ist in Abbildung 2.3 zu sehen.

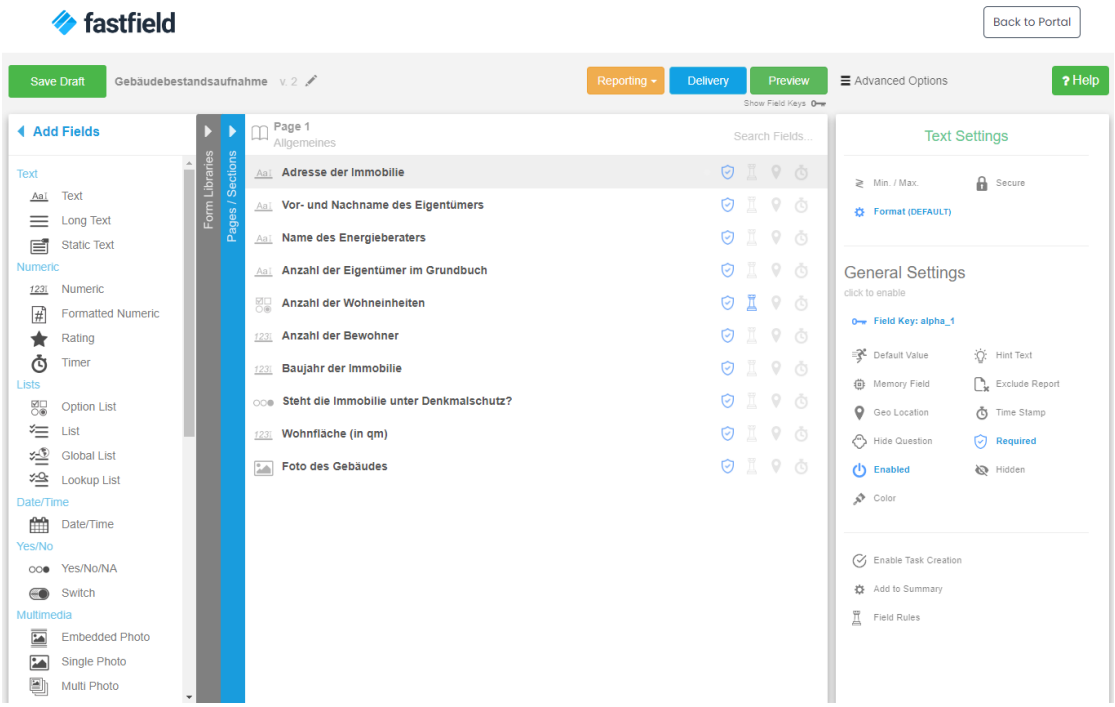


Abbildung 2.3: Erstellung eines Formulars mit der FastField Web-App

Die mobile App ermöglicht dann das Ausfüllen eines zuvor angelegten Formulars, ein Screenshot der Benutzungsschnittstelle ist in Abbildung 2.4 zu sehen. Dabei können die Felder der Abschnitte ausgefüllt werden, Anmerkungen gemacht und schließlich das Formular abgesendet werden. Sofern das Formular offline ausgefüllt wurde, wird es synchronisiert, sobald eine Internetverbindung besteht, und ist dann auch über die Web-App einsehbar.

Fazit und Abgrenzung

Grundsätzlich bietet die „FastField“ App viele Funktionen zur Formularerstellung und Ausfüllung und ist, wie die zu entwickelnde App, auch offline nutzbar. Im Vergleich zu der zuvor betrachteten App, die speziell für die Dokumentation von Objektbegehungen konzipiert ist, handelt es sich bei „FastField“ um eine Softwarelösung für allgemeine Inspektionen und Checklisten in verschiedenen Branchen. Allerdings zeigen sich bei genauerer Betrachtung entscheidende Unterschiede und Limitierungen im Vergleich zu den spezifischen Anforderungen des Praxispartners.

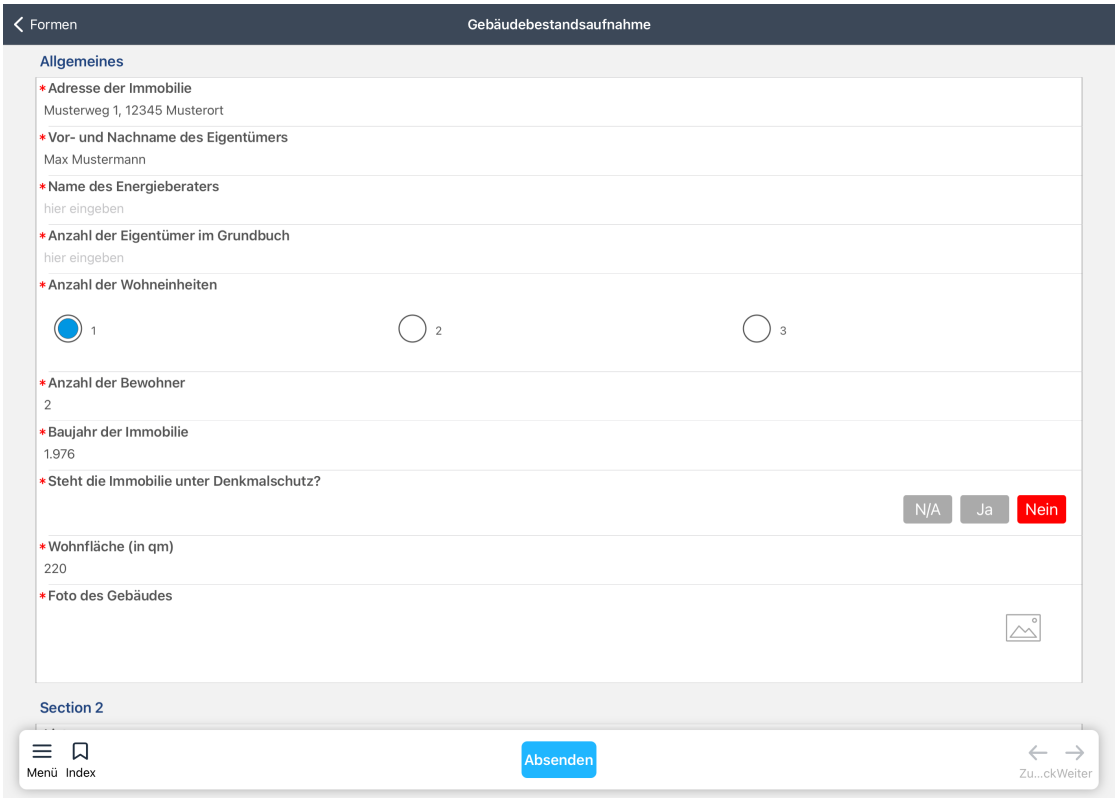


Abbildung 2.4: Formularausfüllung mit der FastField iOS-App

Auch wenn die Erstellung von Gebäudebestandsaufnahmen über Formulare die Hauptanforderung der zu entwickelnden App ist, benötigt der Praxispartner eine maßgeschneiderte Softwarelösung, die über die reine Formularerstellung und Ausfüllung hinausgeht. Insbesondere ist eine tiefgreifende Integration von Kunden- und Projektdaten erforderlich. Die zu entwickelnde App soll nicht nur Bestandsaufnahmen erfassen, sondern diese Daten auch direkt mit den jeweiligen Kunden- und Projektinformationen verknüpfen. Dies ermöglicht eine effizientere Verwaltung und Nachverfolgung der Projekte und sorgt für eine konsistente und nachvollziehbare Dokumentation aller erhobenen Daten.
Zudem muss die zukünftige Softwarelösung spezifische Arbeitsprozessanforderungen berücksichtigen, um optimal in die bestehenden Arbeitsabläufe integriert zu werden. Eine generische Lösung wie „FastField“ kann diese spezifischen Anforderungen nicht in der notwendigen Tiefe abbilden. Die Eigenentwicklung erlaubt es hingegen, die App genau auf die Arbeitsweise und Bedürfnisse des Praxispartners zuzuschneiden, was zu einer höheren Praxistauglichkeit führt.
Ein weiterer Vorteil der Eigenentwicklung ist die zukünftige Erweiterbarkeit. Der Praxispartner plant, die App kontinuierlich zu erweitern, um auf sich ändernde Anforderungen reagieren zu können. Beispielsweise ist angedacht, Teile der App in Zukunft auch für Kunden oder externe Gewerke zur Verfügung zu stellen. Eine proprietäre Lösung wie „FastField“ bietet nicht die gleiche Flexibilität und Anpassungsfähigkeit, da Änderungen und Erweiterungen von den Vorgaben und dem

Entwicklungstempo des externen Softwareanbieters abhängig sind.
Insgesamt ist die Eigenentwicklung einer Software entscheidend, um die spezifischen Anforderungen des Praxispartners optimal zu erfüllen, und damit eine hohe Tauglichkeit im Praxiseinsatz sicherzustellen sowie die zukünftige Erweiterbarkeit offenzuhalten.

3

Grundlagen

Im folgenden Kapitel werden Grundbegriffe und Konzepte erläutert, die für das Verständnis dieser Arbeit relevant sind.

3.1 Gebrauchstauglichkeit

Der Begriff der Gebrauchstauglichkeit wird gemäß DIN EN ISO 9241-11 wie folgt definiert:
„Die Gebrauchstauglichkeit ist das Ausmaß, in dem ein System, ein Produkt oder eine Dienstleistung durch bestimmte Benutzer genutzt werden kann, um in einem bestimmten Nutzungskontext bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [ISO18].
Dabei handelt es sich, wie in Abbildung 3.1 zu sehen, um das Ergebnis der Nutzung, welches von den Zielen, der Art der Benutzer und weiteren Komponenten des Nutzungskontextes abhängt. Der Nutzungskontext wird als die „Kombination von Benutzern, Zielen, Aufgaben, Ressourcen und Umgebung“ definiert. [ISO18]
Die Verwendung desselben Systems, Produkts oder derselben Dienstleistung kann daher je nach Nutzungskontext eine signifikant unterschiedliche Gebrauchstauglichkeit ergeben. [ISO18]

3.2 Nutzerzentrierter Entwicklungsprozess

Der in dieser Arbeit verwendete nutzerzentrierte Entwicklungsprozess basiert auf der in ISO-Norm 9241-210 definierten menschzentrierten Gestaltung interaktiver Systeme.
Diese hat das Ziel, eine verbesserte Gebrauchstauglichkeit interaktiver Systeme zu erreichen, indem sich bei der Gestaltung und Entwicklung auf die Benutzer, deren Erfordernisse und Anforderungen konzentriert wird. Bei menschzentrierter Gestaltung wird im Gegensatz zu benutzerzentrierter Gestaltung betont, dass neben den Nutzern auch Stakeholder berücksichtigt werden.[ISO20]
Bei diesem Prozess sollen Unsicherheiten mithilfe von Iteration schrittweise beseitigt werden, indem „Beschreibungen, Spezifikationen und Prototypen überarbeitet

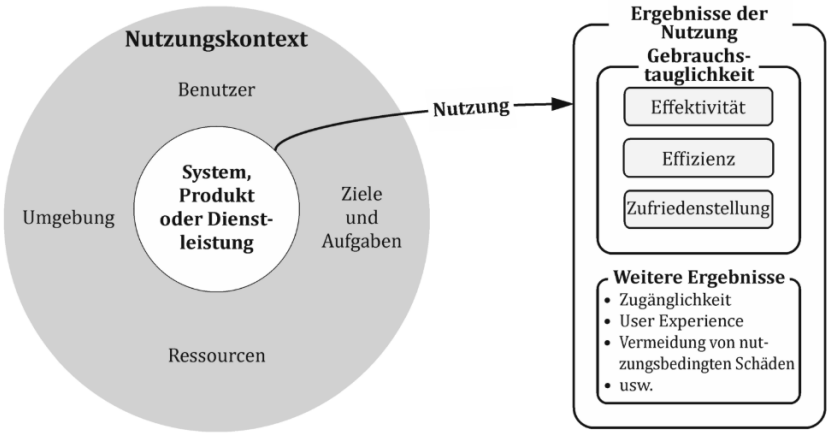


Abbildung 3.1: Gebrauchstauglichkeit als Ergebnis der Nutzung in einem bestimmten Nutzungskontext [ISO18]

und verbessert werden, sobald neue Informationen gewonnen wurden“.[ISO20] Die menschzentrierte Gestaltung sieht, wie in Abbildung 3.2 dargestellt, vier miteinander verbundene Gestaltungsaktivitäten vor: das „Verstehen und Beschreiben des Nutzungskontexts“, das „Spezifizieren der Nutzungsanforderungen“, das „Erarbeiten von Gestaltungslösungen“ und das „Evaluieren der Gestaltung“. [ISO20]

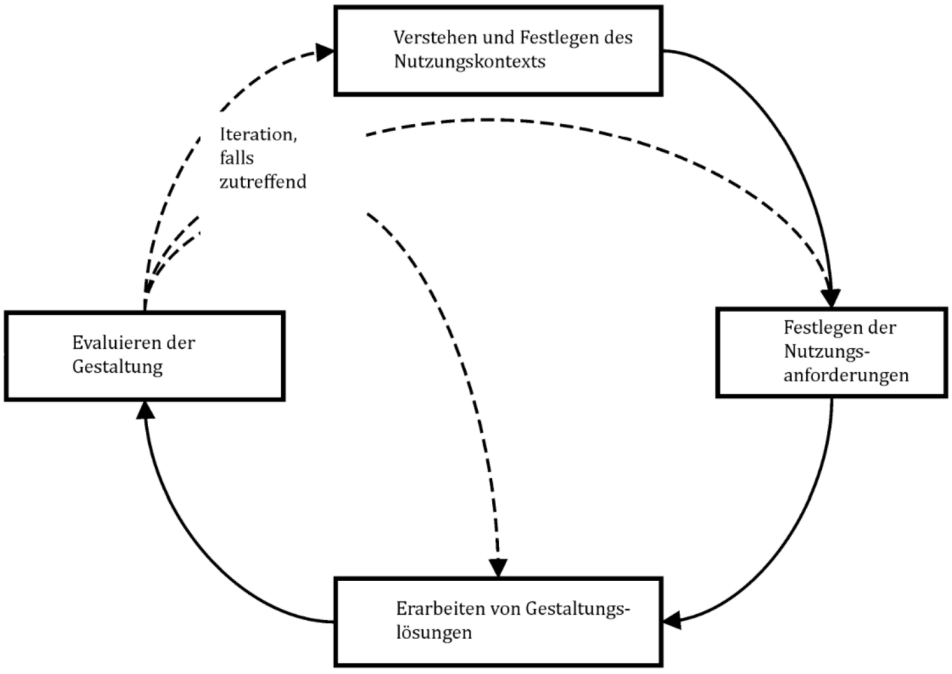


Abbildung 3.2: Wechselseitige Abhängigkeit menschzentrierter Gestaltungsaktivitäten [ISO20] (vereinfacht)

3.3 Technische Gebäudebestandsaufnahme

Eine technische Gebäudebestandsaufnahme ist ein systematischer Arbeitsablauf, bei dem durch einen Mitarbeiter die technischen Details zu den Bauteilen eines Wohnhauses vor Ort erfasst und dokumentiert werden. Bauteile umfassen beispielsweise das Dach, die Fenster, die Heizungsanlagen etc. Diese detaillierte Erfassung und Dokumentation ist essenziell, um einen umfassenden Überblick über den Zustand und die Beschaffenheit eines Gebäudes zu erhalten und mögliche Schwachstellen zu identifizieren.
Eine technische Gebäudebestandsaufnahme zielt darauf ab, spezifische Gebäudedaten für eine spätere Verarbeitung zu sammeln, beispielsweise zur Erstellung eines individuellen Sanierungsfahrplans.

3.4 Individueller Sanierungsfahrplan

Der individuelle Sanierungsfahrplan (iSFP) ist ein vom Bundesamt für Wirtschaft und Ausfuhrkontrolle (BAFA) gefördertes Instrument, das Eigenheimbesitzern eine strategische und schrittweise Anleitung zur energetischen Sanierung ihrer Immobilie bietet.[BAF]
Dieser Plan wird durch einen Energieeffizienzexperten erstellt und enthält aufeinander aufbauende Maßnahmenpakete, die aufzeigen, wie das Gebäude über einen längeren Zeitraum umfassend energetisch saniert werden kann. Das Ziel dabei ist die Verringerung des Energiebedarfs und damit eine Einsparung von CO_2.[BAF]

4

Analyse

Um eine Software zur Optimierung des Prozesses der technischen Gebäudebestandsaufnahme zu entwickeln, die den Nutzungsanforderungen entspricht, ist zunächst ein Verständnis des Nutzungskontextes erforderlich. Daher wird im Folgenden der aktuelle Arbeitsablauf beschrieben und analysiert.
Diese Untersuchung wurde durch eine Reihe von Feldstudien durchgeführt, bei denen die Mitarbeiter während ihres Arbeitsprozesses begleitet, beobachtet und befragt wurden. Insgesamt wurden drei verschiedene Mitarbeiter, die Energieeffizienzexperten sind, teilweise mehrmals, bei Vor-Ort-Kundenterminen zur Datenerfassung begleitet und auch mit den Mitarbeitern zur Projektbearbeitung im Büro gesprochen.
Diese Vorgehensweise ermöglicht ein tieferes Verständnis der Tätigkeiten der Mitarbeiter sowie der Gründe für ihre Handlungsweisen.

4.1 Aktueller Arbeitsablauf

4.1.1 Beschreibung

Technische Gebäudebestandsaufnahmen werden bei der Firma TRAGWERK insbesondere dann durchgeführt, wenn ein individueller Sanierungsfahrplan (iSFP) für ein Gebäude erstellt werden soll. In den meisten Fällen handelt es sich dabei um Wohngebäude, wie Einfamilienhäuser oder Mehrfamilienhäuser, die saniert werden sollen, um deren Energieeffizienz zu verbessern und somit die Heizkosten für die Eigentümer zu senken.
Diese Arbeit fokussiert sich auf die Datenerfassung für das Erstellen von individuellen Sanierungsfahrplänen für bestehende Wohngebäude. Dabei besteht der Prozess aus mehreren aufeinanderfolgenden Teilschritten, die meist von verschiedenen Mitarbeitern durchgeführt werden. Diese Teilschritte sind in Diagramm 4.1 dargestellt und werden im Folgenden beschrieben.

Kundengespräch zur Erfassung der Projektdetails

Der Arbeitsablauf beginnt mit dem Eingang eines neuen Auftrags zur Erstellung eines iSFP im Büro. Das Kundengespräch wird meist von einem bestimmten Mitarbeiter, der für die Kundenbetreuung zuständig ist, durchgeführt. Zunächst werden

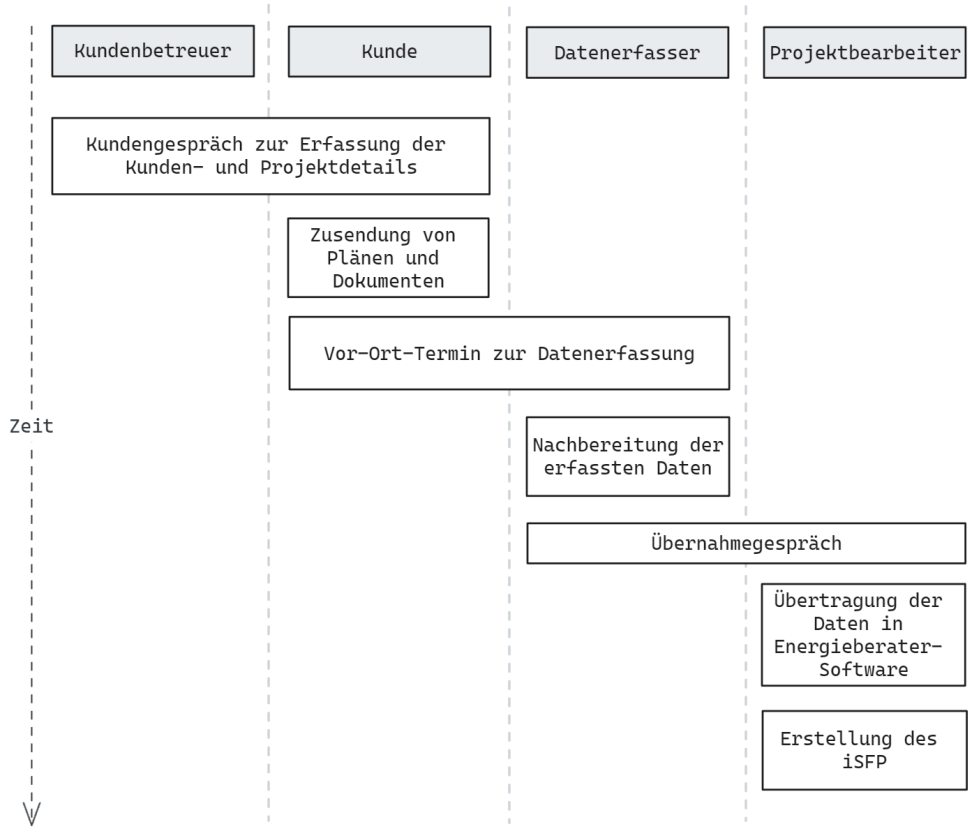
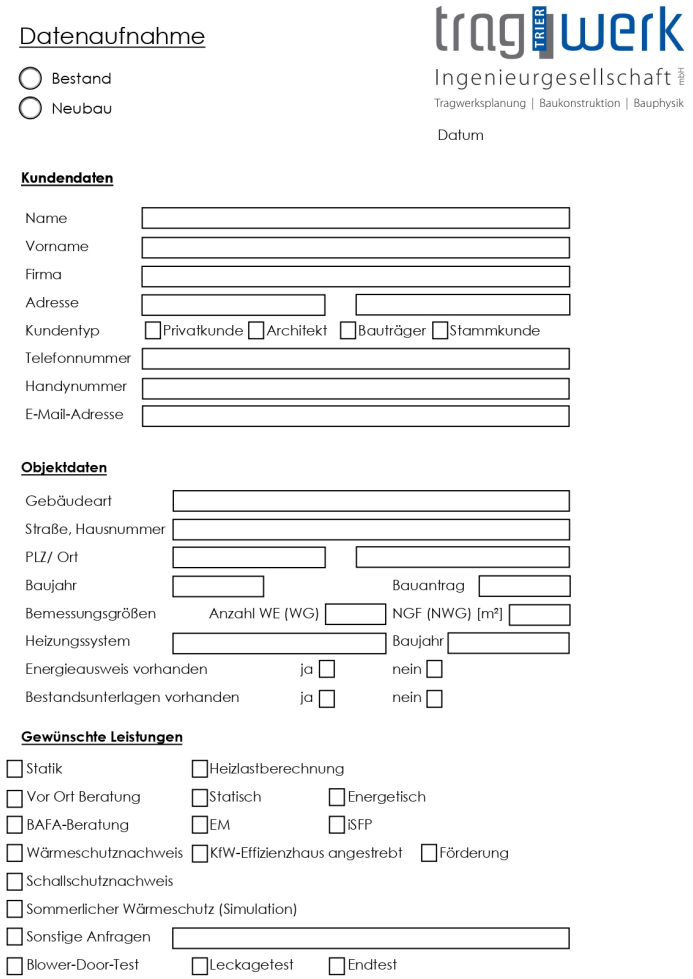


Abbildung 4.1: Teilschritte der verschiedenen Akteure bei der Erstellung eines iSFP

am Computer grundlegende Daten zum Kunden, Gebäude und den gewünschten Leistungen in einer Excel-Tabelle erfasst. Das entsprechende Formular ist in Abbildung 4.2 dargestellt. Das ausgefüllte Dokument wird anschließend im firmeninternen Dateinetzwerk in einem neu angelegten Projektorder gespeichert und ist somit für alle Mitarbeiter zugänglich. Im Verlauf dieses Kundegesprächs wird auch ein Vor-Ort-Termin zur Datenerfassung vereinbart und der Kunde darum gebeten, Dokumente wie Pläne, Grundrisse, Gutachten usw. zum Gebäude per E-Mail zuzusenden.

Vor-Ort-Termin zur Datenerfassung

Anschließend erfolgt der Vor-Ort-Termin zur Gebäudedatenerfassung mit dem Kunden, der von einem zertifizierten Energieeffizienzexperten, durchgeführt wird. Dieser Termin beginnt mit einem Gespräch mit dem Kunden und möglichen Mietern, bei dem zunächst die angedachten Umbaumaßnahmen besprochen und dann einige grundlegende Daten abgefragt werden, z. B. die Bauweise oder Heizart. Dabei wird eine eigens entwickelte Excel-Tabellen-Vorlage (siehe Abbildung 4.3) verwendet, um alle relevanten Daten handschriftlich zu notieren. Diese wird auf einem iPad geöffnet, um die einzelnen Felder handschriftlich mithilfe des Apple



Datenaufnahme

Bestand
Neubau

tragwerk TRIER Ingenieurgesellschaft mbH
Tragwerksplanung | Baukonstruktion | Bauphysik

Datum

Kundendaten

Name
Vorname
Firma
Adresse
Kundentyp ☐Privatkunde ☐Architekt ☐Bauträger ☐Stammkunde
Telefonnummer
Handynummer
E-Mail-Adresse

Objektdaten

Gebäudeart
Straße, Hausnummer
PLZ/ Ort
Baujahr Bauantrag
Bemessungsgrößen Anzahl WE (WG) NGF (NWG) [m²]
Heizungssystem Baujahr
Energieausweis vorhanden ja ☐ nein ☐
Bestandsunterlagen vorhanden ja ☐ nein ☐

Gewünschte Leistungen

☐Statik ☐Heizlastberechnung
☐Vor Ort Beratung ☐Statisch ☐Energetisch
☐BAFA-Beratung ☐EM ☐iSFP
☐Wärmeschutznachweis ☐KfW-Effizienzhaus angestrebt ☐Förderung
☐Schallschutznachweis
☐Sommerlicher Wärmeschutz (Simulation)
☐Sonstige Anfragen
☐Blower-Door-Test ☐Leckagetest ☐Endtest

Abbildung 4.2: Formular zur Datenaufnahme eines neuen Projekts

Pencils mit Inhalt zu füllen. Alternativ wird das Formular ausgedruckt und in Papierform ausgefüllt.
Es folgt ein Rundgang durch alle Etagen und Räume des Wohngebäudes sowie eine Begutachtung von Außen. Dabei werden neben den textuellen Daten auch Fotos aufgenommen. Dies geschieht zum einen mit dem Smartphone, mit welchem Bauteile wie Fenster und die Anlagentechnik und deren Typenschildern fotografiert und zum anderen mit einer Drohne, mit der Bilder von den Fassaden und dem Dach des Gebäudes gemacht werden. Die Drohne wird dabei per Smartphone gesteuert und die Bilder auf diesem lokal gespeichert. Von besonderem Interesse beim Rundgang durch das Haus sind die Fenster, Dämmungen und Heizinstallationen.
Abgeschlossen wird der Termin mit der Frage danach, welche Wünsche und Anforderungen der Kunde hat, beispielsweise welche Sanierungsmaßnahmen angedacht sind oder auf jeden Fall durchgeführt werden.

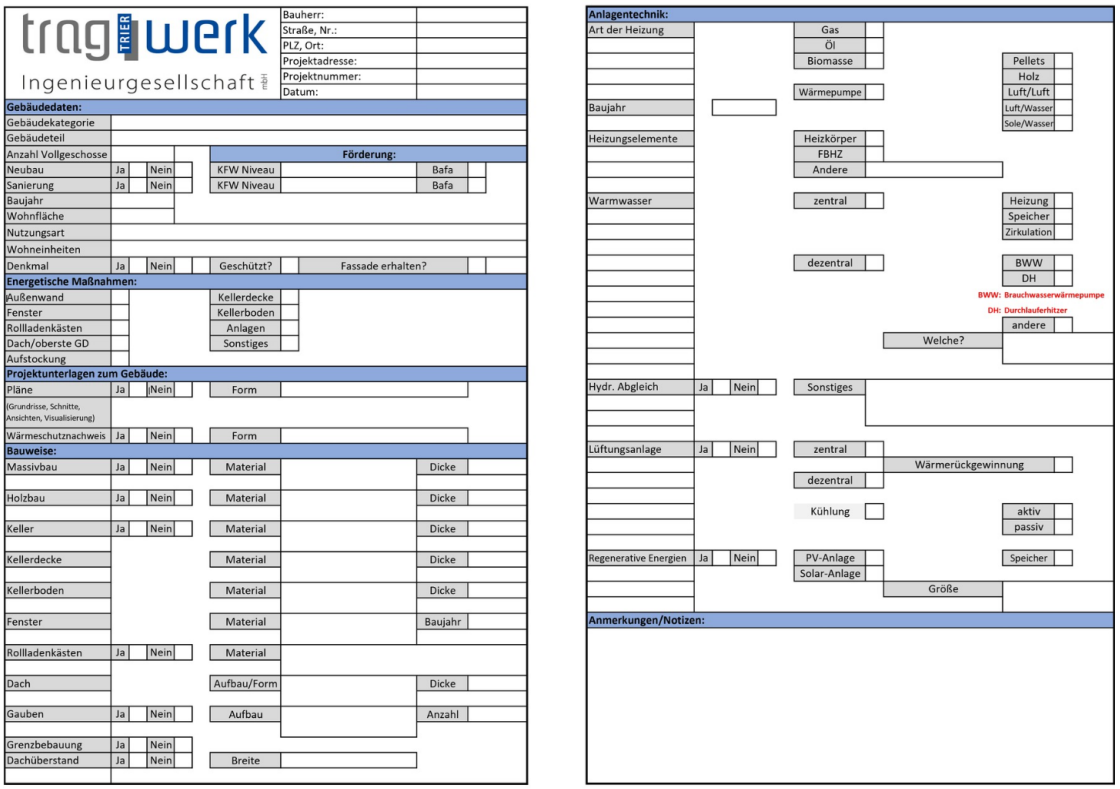


Abbildung 4.3: Formular für die Bestandsaufnahme bei einem Vor-Ort-Termin

Ein Vor-Ort-Termin dauert je nach Größe des Gebäudes in der Regel, ohne Einbeziehung der Anfahrtszeit, zwischen ein und zwei Stunden.

Nachbereitung der erfassten Daten

Nach Abschluss des Vor-Ort-Termins findet die Nachbereitung der handschriftlich erfassten Daten im Büro statt. Dabei werden zunächst alle Bilder (meist 20–50 Stück) vom Smartphone auf den PC übertragen und im Projektorder im Dateinetzwerk gespeichert. Außerdem werden die handschriftlichen Notizen vom iPad am Computer in eine neue, leere, Datenerfassungsvorlage abgetippt.
Dieser Vorgang dauert meist zwischen 15 und 30 Minuten.

Übertragung der erfassten Daten in die Energieberater-Software

Gefolgt von der Nachbereitung findet, falls die Datenerfassung und Projektbearbeitung durch unterschiedliche Mitarbeiter erfolgt, ein Übernahmegespräch zwischen dem Mitarbeiter zur Datenerfassung und jenem, der aus den Daten einen Sanierungsfahrplan erstellt, im folgenden Projektbearbeiter genannt, statt. In diesem Gespräch werden die Kundenwünsche oder sonstige relevante Bemerkungen zu den erfassten Daten besprochen.
Die Projektbearbeiter benutzen für Ihre Arbeit die Desktop-Energieberater-Software

„ZUB Helena“ der ZUB Systems GmbH. Mithilfe dieser Energieberater-Software wird das Gebäude modelliert, die Maßnahmen geschnürt und letztlich der Sanierungsfahrplan erstellt.

Begonnen wird mit der Übertragung der Projekt- und Kundendaten aus dem im Dateinetzwerk gespeicherten PDF-Dokument zur Projektaufnahme. Abbildung 4.4 zeigt die Software bei der Eingabe der Daten zum Auftraggeber.

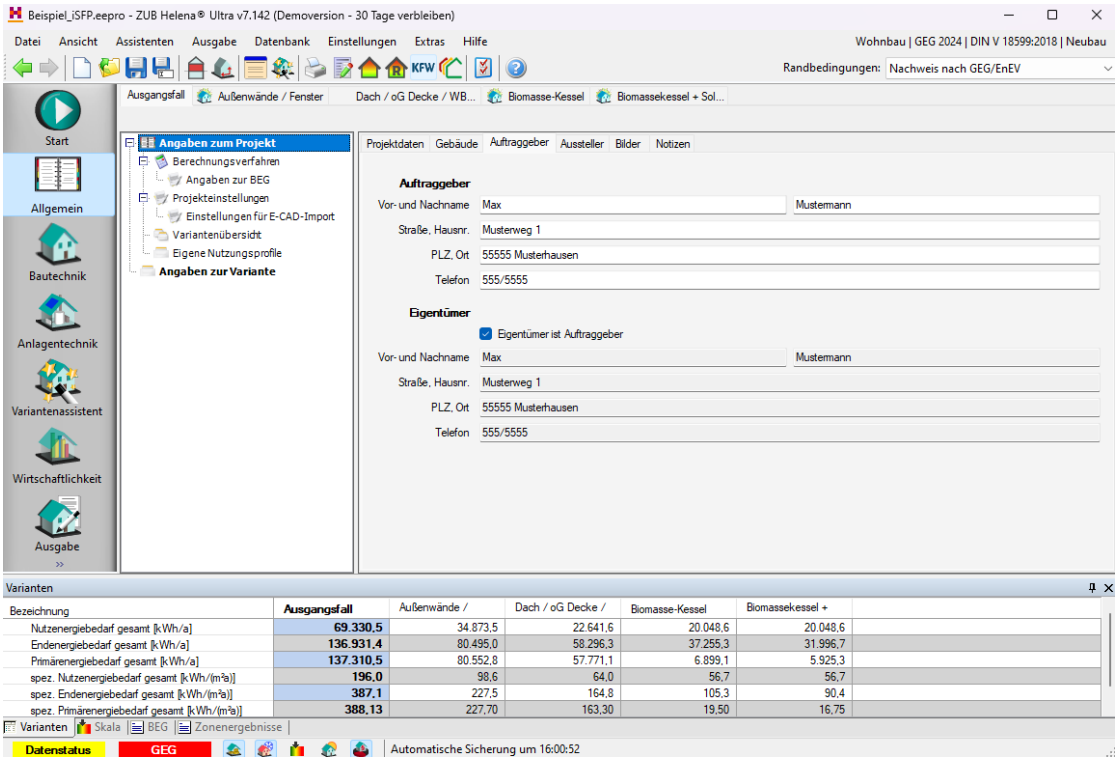


Abbildung 4.4: Screenshot der Software ZUB Helena beim Eintragen der Daten zum Auftraggeber

Danach wird das PDF-Dokument der Vor-Ort-Bestandsaufnahme geöffnet und die erhobenen Daten in die Software eingetragen. Dabei werden auf Basis des Baujahrs und Typenschildern Wärmeleitwerte und weitere relevante Kennzahlen ermittelt. Falls nötig, werden Pläne und Dokumente zum Gebäude zur Hilfe genommen, um alle relevanten Daten herauszufinden und anzugeben.
Abbildung 4.5 zeigt exemplarisch die Eingabe der Heizung, die unter der Oberkategorie Anlagentechnik aufgelistet ist.
Daraufhin werden Berechnungen ausgeführt und die Sanierungsmaßnahmenpakete erstellt. Dieser Ablauf ist für diese Arbeit nicht von Bedeutung und wird daher nicht weiter beschrieben.

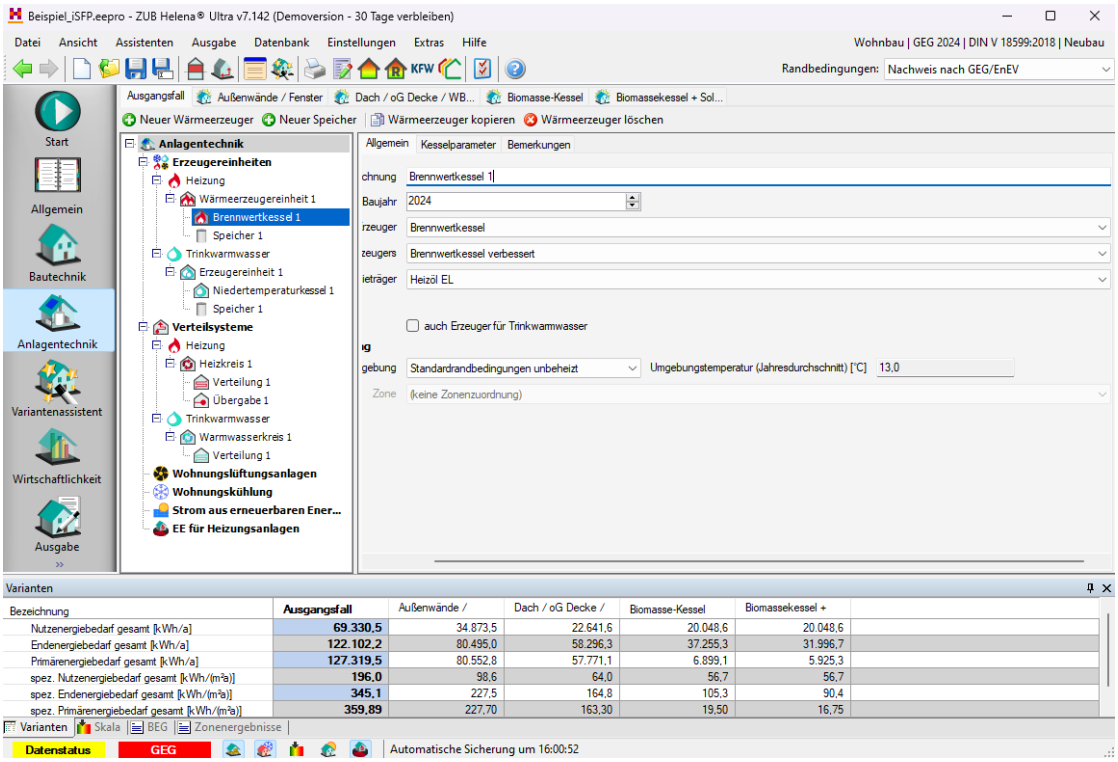


Abbildung 4.5: Screenshot der Software ZUB Helena beim Eintragen der Daten zur Heizung

Erstellung des iSFP mithilfe der Energieberater-Software

Nachdem alle relevanten Daten in das Programm eingetragen wurden, die Berechnungen durchgeführt und die Sanierungsmaßnahmenpakete erstellt wurden, kann der individuelle Sanierungsfahrplan in PDF-Form generiert werden.
Dabei kommt die Sanierungsfahrplan-Funktion (siehe Abbildung 4.6) der Software zum Einsatz. Diese Funktion fügt die zuvor erzeugten Daten inklusive zusätzlicher Bildern und Texte in eine Sanierungsfahrplan-Vorlage ein, die vom Programm bereitgestellt wird.
Das daraus resultierende Dokument, welches der fertige iSFP ist, wird nun dem Kunden zur Verfügung gestellt. Die beiden Teilschritte, bei denen die Energieberater-Software zum Einsatz kommt, dauern in der Regel einen Arbeitstag.

4.1.2 Vorteile und Stärken

Die Vorteile und Stärken des aktuellen Prozesses sollen in Zukunft, wenn möglich, erhalten bleiben und sind daher ebenso wie die Probleme und Schwächen ein Aspekt, der betrachtet wird.
Ein Hauptvorteil des aktuellen Verfahrens ist die Flexibilität bei der Datenerfassung vor Ort. Da die Daten komplett handschriftlich erfasst werden, ist es leicht möglich, Sonderfälle zu notieren, die von der Formularvorlage nicht abgedeckt sind.

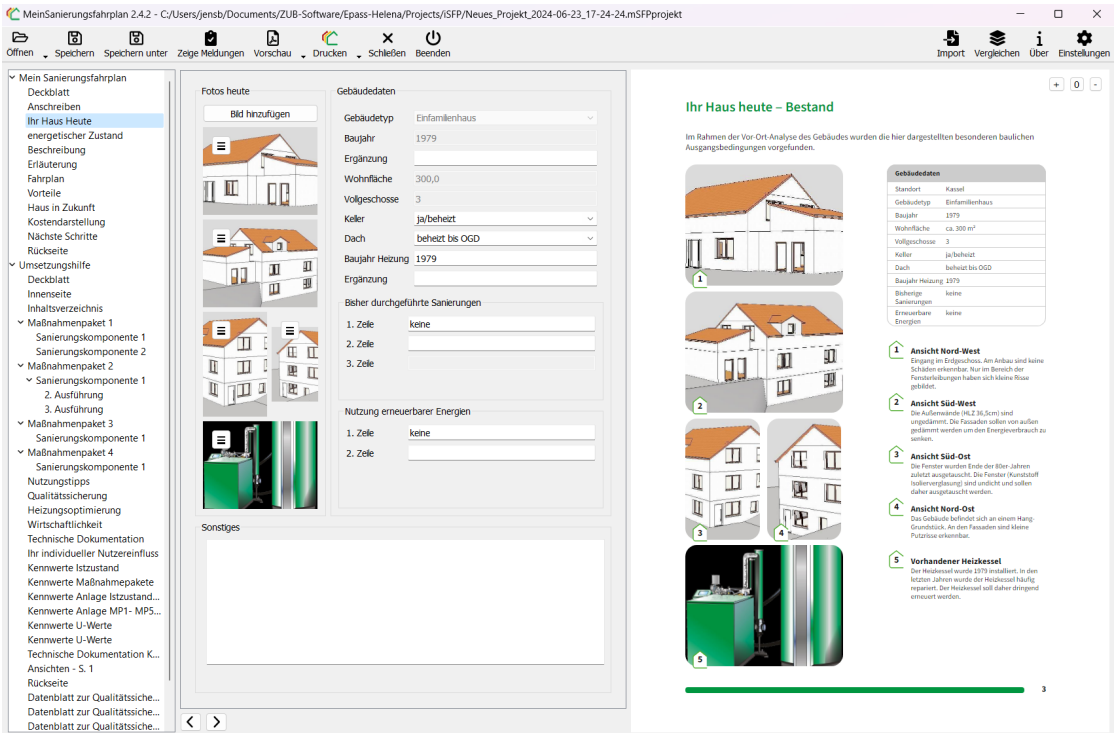


Abbildung 4.6: Screenshot der Software ZUB Helena beim Eintragen der Daten zur Heizung

Ein weiterer Vorteil liegt in der zentralen Datenablage. Der Projektordner im Dateilaufwerk der Firma kann zur Ablage der ausgefüllten Formulare, Kundendokumente, Bilder und der Projektdatei der Energieberater-Software genutzt werden. Alle Daten sind somit an einem Ort für alle Mitarbeiter zugänglich.

4.1.3 Probleme und Schwächen

Trotz der genannten Vorteile gibt es einige signifikante Probleme und Schwächen im aktuellen Prozess, die die Effizienz der Datenerfassung und Projektbearbeitung beeinträchtigen. Diese Schwächen müssen identifiziert und adressiert werden, um eine Verbesserung des gesamten Arbeitsablaufs zu erreichen.

Variable Qualität und Vollständigkeit der Bestandsaufnahme

Ein zentrales Problem der bisherigen Methode ist, dass laut Projektbearbeitern die erfassten Daten in ihrer Qualität und Vollständigkeit variabel sein können. Gelegentlich kommt es vor, dass einige wichtige Daten beim Vor-Ort-Termin nicht aufgenommen werden, weshalb im schlimmsten Fall ein erneuter Termin ausgemacht werden muss. Darüber hinaus führen Lücken in den Daten dazu, dass Werte pauschalisiert werden müssen oder der Projektbearbeiter Baudokumente durchsuchen muss, um die benötigten Angaben herauszufinden, was zeitaufwändig ist.

Nachbereitung der erfassten Daten

Ein weiterer Nachteil besteht darin, dass die handschriftlich erfassten Formulare des Vor-Ort-Termins im Rahmen der Nachbereitung im Büro am PC in Textform erneut aufgeschrieben werden müssen. Zudem müssen auch die aufgenommenen Fotos manuell vom Smartphone auf den PC übertragen werden. nochmal Dies führt neben einem erhöhten Zeitaufwand auch zu einer höheren Wahrscheinlichkeit für Fehler bei der Übertragung.

Redundanz der Kundendaten

Ein weiteres Problem am aktuellen Prozess ist die mehrfache Notwendigkeit der Angabe der Kundendaten. Zunächst werden diese beim Erstgespräch in einem Excel-Formular am PC erfasst, anschließend erneut handschriftlich während der Bestandsaufnahme vor Ort auf dem benutzten Formular notiert und während der Nachbereitung erneut in das Formular zur Bestandsaufnahme eingetragen. Diese Redundanz erhöht das Risiko von Inkonsistenzen und Fehlern.

Unstrukturierte Daten

Im aktuellen Prozess der Gebäudebestandsaufnahme werden die Daten unstrukturiert in verschiedenen Formaten wie PDFs und Excel-Tabellen gespeichert. Diese unstrukturierte Datenspeicherung verhindert die Integration von anderen Softwaresystemen. Ohne eine einheitliche Datenstruktur ist es nicht möglich, Schnittstellen zu anderen Systemen zu etablieren, was notwendig wäre, um Prozesse wie die Integration eines Customer-Relationship-Management (CRM) zu ermöglichen, was der Praxispartner in Zukunft nicht ausschließt.

Informationsverlust bei Arbeitsteilung

In der Vergangenheit hat meist derselbe Mitarbeiter die Daten beim Vor-Ort-Termin erfasst und anschließend im Büro den iSFP erstellt. Diese Vorgehensweise gewährleistet eine durchgängige Informationskette und ermöglicht es dem Mitarbeiter, Kundenwünsche und spezifische Anforderungen direkt in den iSFP einfließen zu lassen.
Zukünftig sollen jedoch vermehrt spezialisierte Mitarbeiter die Datenerfassung vor Ort übernehmen, während andere Mitarbeiter im Büro den iSFP erstellen. Ein Problem dabei ist, dass die Mitarbeiter im Büro nur die Informationen berücksichtigen können, die schriftlich dokumentiert oder im Übergabegespräch vermittelt wurden. In der Vergangenheit führte dies bereits vereinzelt dazu, dass wichtige Kundenwünsche und spezifische Anforderungen, die vor Ort besprochen wurden, nicht in den iSFP einflossen. Auch wenn dieser Fehler bei einer Kontrolle vor der Fertigstellung des iSFP entdeckt wurde, verursachte dies dennoch einen Mehraufwand für die Mitarbeiter.

4.2 Nutzungskontextanalyse

Um eine Softwarelösung mit hoher Gebrauchstauglichkeit zu entwickeln, wird im Folgenden der Nutzungskontext auf Basis der zuvor gesammelten Erkenntnisse ausgearbeitet. Dieser Schritt ermöglicht es, die spezifischen Eigenschaften der verschiedenen künftigen Benutzergruppen genau zu definieren um daraus Nutzungsanforderungen abzuleiten.
Die verschiedenen Nutzergruppen der zukünftigen Software wurden rollenbasiert aufgeteilt, um den spezifischen Anforderungen und Verantwortlichkeiten der unterschiedlichen Benutzergruppen gerecht zu werden. Generell zählen zu den Nutzern der App ausschließlich Mitarbeiter der Ingenieurgesellschaft TRAGWERK. Da das Unternehmen bereits zahlreiche digitale Arbeitsabläufe nutzt, sind die Mitarbeiter routiniert im Umgang mit Geräten wie PCs und Tablets.

4.2.1 Datenerfasser

Die primäre Nutzergruppe der App besteht aus den Mitarbeitern, die die technische Gebäudebestandsaufnahme bei den Vor-Ort-Terminen durchführen. Hierzu zählen ausschließlich zertifizierte Energieeffizienzexperten.
Diese Personen verfügen über ein hohes Verständnis über Gebäudebauteile und Energieeffizienz und haben bereits bei vielen verschiedenen Wohngebäuden Bestandsaufnahmen durchgeführt. Ihr umfassendes Domänenverständnis bedeutet, dass die App ihre Fachsprache verwenden muss, um effektiv genutzt werden zu können. Da sie bereits routiniert iPads für die Bestandsaufnahme einsetzen, kann ein hohes Maß an technischem Verständnis vorausgesetzt werden.
Die Hauptaufgabe dieser Nutzergruppe besteht in der Erstellung der Bestandsaufnahme vor Ort. Das Ziel dabei ist es, alle für den iSFP relevanten technischen Daten und die Kundenwünsche zu erfassen, sowie dem Kunden eventuelle Fragen zur Energieeffizienz und zum iSFP zu beantworten.
Die physische Umgebung, in der die Datenerfasser arbeiten, umfasst bestehende Wohngebäude, die energetisch saniert werden sollen. Diese Gebäude können sich in verschiedenen Zuständen befinden, von unbewohnten, sich im Umbau befindenden Häusern bis zu bewohnten Gebäuden. Die Arbeitsumgebung innerhalb dieser Gebäude variiert stark und kann Wohnräume, Heiz- und Technikräume, Keller sowie Außenbereiche umfassen. Diese Umgebungen können zu eingeschränkten Sichtverhältnissen und Bewegungsmöglichkeiten führen.
Die soziale Umgebung ist durch die direkte Interaktion mit den Gebäudebesitzern geprägt. Einerseits müssen die Datenerfasser technische Daten und Merkmale erfassen, andererseits im Gespräch mit den Kunden bleiben. Diese doppelte Herausforderung erfordert sowohl technisches Fachwissen als auch ausgeprägte kommunikative Fähigkeiten, um vertrauensvolle Beziehungen zu den Kunden aufzubauen und ihre Bedürfnisse zu verstehen.
Für ihre Arbeit nutzen die Datenerfasser verschiedene Arbeitsmittel. Zur Aufnahme von Bildern der Bauteile oder Typenschildern verwenden sie ein Smartphone, da dieses handlich ist und über ein Weitwinkelkameraobjektiv verfügt. Für Formulare und Checklisten kommen 10,9 Zoll große iPads mit Stift zum Einsatz, die

aufgrund ihrer Größe und Funktionalität geeignet zum Ausfüllen von Formularen sind. Diese iPads verfügen über keine mobile Datenflat und haben daher nur im Büro eine Internetverbindung. Außenaufnahmen des Gebäudes werden mithilfe von Drohnen durchgeführt. Zusätzlich nutzen sie verschiedene Messwerkzeuge wie Meterstäbe und Lasermessgeräte, um genaue Messungen vorzunehmen.

4.2.2 Projektbearbeiter

Eine weitere Nutzergruppe der App sind die Mitarbeiter, die aus den erfassten Daten der Vor-Ort-Termine im Büro einen Sanierungsfahrplan erstellen. Diese Gruppe überschneidet sich teilweise mit der der Datenerfasser, jedoch gibt es auch Mitarbeiter, die ausschließlich Projekte bearbeiten und keine Vor-Ort-Bestandsaufnahmen durchführen.
Die Mitarbeiter dieser Gruppe verfügen ebenso wie die Datenerfasser über ein hohes Verständnis im Bereich der Gebäudebauteile und Energieeffizienz. Sie besitzen fundierte Kenntnisse in der Bewertung und Analyse von Gebäudedaten sowie der Erstellung von geeigneten Sanierungsmaßnahmen.
Die Hauptaufgabe dieser Nutzergruppe besteht darin, aus den erfassten Daten einen individuellen Sanierungsfahrplan zu erstellen. Dazu müssen sie zunächst die erfassten Daten in die spezielle Energieberater-Software übertragen und mit Daten aus Baudokumenten ergänzen. Danach folgt die Erstellung der Sanierungsmaßnahmen und des Sanierungsfahrplans. Dabei ist es wichtig, die Wünsche und Bedürfnisse der Auftraggeber zu berücksichtigen.
Die physische Umgebung während dieses Arbeitsablaufs ist das Firmenbüro. Die Mitarbeiter arbeiten an ihren Schreibtischen in einer Büroumgebung, die auf konzentrierte und analytische Tätigkeiten ausgelegt ist.
Die soziale Umgebung der Projektbearbeiter ist geprägt durch die Interaktion mit Kollegen im Büro. Obwohl der Großteil der Arbeit eigenständig am Schreibtisch erledigt wird, gibt es gelegentlich Abstimmungen und Rückfragen mit dem Kollegen, der die Datenerfassung durchgeführt hat, insbesondere bei Unklarheiten oder fehlenden Informationen.
Für ihre Arbeit nutzen sie einen Desktop-PC mit zwei Monitoren, um gleichzeitig die erfassten Gebäudedaten und die Energieberater-Software zur Erstellung der Sanierungsfahrpläne anzeigen zu können. Zusätzlich greifen sie auf Baudokumente und andere relevante Unterlagen zu, die digital vorliegen. Kommunikationsmittel wie E-Mail und Telefon sind ebenfalls essenziell, um Rückfragen zu klären und Informationen mit Kollegen auszutauschen.

4.2.3 Weitere Benutzer

Neben diesen beiden primären Nutzergruppen soll die zukünftige App auch zur Erfassung der Projekt- und Kundendaten beim Erstgespräch genutzt werden. Dies geschieht entweder durch einen Mitarbeiter, der im Unternehmen für die Kundenbetreuung zuständig ist und ebenso zur Gruppe der Datenerfasser gehört, oder alternativ durch den Mitarbeiter, der den Vor-Ort-Termin zur Bestandsaufnahme

durchführt. Da die Nutzungskontexte viele Überschneidungen haben, wird die Erfassung der Projekt- und Kundendaten als Teil der Datenerfassung angesehen.
Eine weitere Benutzergruppe sind Administratoren, die bei Bedarf Benutzerkonten für Mitarbeiter hinzufügen oder löschen können. Dieser Gruppe gehört der Geschäftsführer der Firma TRAGWERK und der App-Entwickler an. Da die Benutzerverwaltung kein Kernpunkt dieser Arbeit ist und eine aus Nutzerperspektive vergleichsweise simple Funktion darstellt, wird diese Nutzergruppe im Folgenden nicht weiter betrachtet.

4.3 Nutzungsanforderungen

Auf Basis der zuvor analysierten Nutzungskontexte und identifizierten Benutzergruppen werden nun die Nutzungsanforderungen definiert. Diese beschreiben die Fähigkeiten, die das zukünftige Softwaresystem erfüllen muss, sodass die Benutzer ihre Aufgaben damit erledigen können.
Diese ergeben sich aus den bereits beschriebenen Problemen und Schwächen des aktuellen Arbeitsablaufs sowie der Beobachtungen und Erkenntnisse im Rahmen der Feldstudien während der Analysephase.
Im Einklang mit dem iterativen Charakter des benutzerzentrierten Entwicklungsprozesses wurden Teile dieser Anforderungen auch nach der initialen Analysephase, etwa während der Evaluation kontinuierlich angepasst und aktualisiert, sobald neue Erkenntnisse gewonnen wurden.

4.3.1 Allgemeine Nutzungsanforderungen

Neben den spezifischen Nutzungsanforderungen für bestimmte Benutzergruppen gibt es auch allgemeine Anforderungen an das zu entwickelnde System, die sich aus der Analyse ergeben und die Wünsche und Anforderungen des Praxispartners berücksichtigen.

Kunden- und Projektverwaltung

Da eine Gebäudebestandsaufnahme immer einem Projekt und Kunden zugeordnet ist, muss die Software ermöglichen, zunächst einen Kunden zu erstellen und diesem Kunden ein Projekt zuzuordnen. Bei den Kunden müssen dabei vor allem der Name, die Kontaktdaten und die Adresse gespeichert werden, und bei dem Projekt die Angebots- oder Projektnummer sowie eine gegebenenfalls abweichende Projektadresse.
Diese Kunden- und Projekteinträge müssen innerhalb der Software suchbar und bearbeitbar sein.
Durch diese Kunden- und Projektverwaltung wird sichergestellt, dass Kundendaten, entgegen dem bisherigen Vorgehen, nur einmal erfasst und gespeichert werden, wodurch Redundanzen vermieden und die Effizienz gesteigert wird. Dadurch wird gewährleistet, dass alle Mitarbeiter auf die gleichen, aktuellen Informationen zugreifen können.

Zentrale Cloud-basierte Datenspeicherung

Damit die Daten für alle Mitarbeiter der Firma über das Internet zugänglich sind, ist es notwendig, dass die Software die Kunden- und Projektdaten in einer zentralen Cloud-Datenbank speichert.
Dabei muss sichergestellt werden, dass nur berechtigte Nutzer, also Mitarbeiter der Firma, Zugriff auf die Daten haben.
Eine Anforderung des Praxispartners hierbei ist die Möglichkeit der späteren Anbindung von externen Systemen wie Customer-Relationship-Management (CRM) Systeme. Dies bedeutet, dass es möglich sein muss, die erfassten und gespeicherten Daten zukünftig bei Bedarf in andere Systeme zu übertragen.

Authentifizierung und Autorisierung

Da es sich um eine firmeninterne Anwendung handelt, dürfen die Funktionen der App und vor allem die Kunden- und Projektdaten ausschließlich durch die Mitarbeiter der Firma zugänglich sein. Deshalb muss die Software über ein Authentifizierungssystem und eine Benutzerverwaltung verfügen.
Zudem muss ein Autorisierungssystem sicherstellen, dass nur ausgewählte Nutzer Funktionen wie das Hinzufügen oder Löschen von Benutzerkonten ausführen können.

4.3.2 Datenerfasser

Neben den zuvor genannten allgemeinen Anforderungen ergeben sich spezifische Nutzungsanforderungen für die Nutzergruppe der Datenerfasser. Diese Anforderungen sind darauf ausgerichtet, die Effizienz und Genauigkeit der Vor-Ort-Datenerfassung zu maximieren und die spezifischen Bedürfnisse dieser Benutzergruppe zu erfüllen.

iPad-basierte Datenerfassung

Die Datenerfassung bei Vor-Ort-Terminen findet auf iPads statt, deshalb muss die Software über eine iOS-Version verfügen, die für iPads optimiert ist. Ebenso sollte die Verwendung des Apple Pencil als Eingabemethode möglich sein, da dieser auch im aktuellen Arbeitsablauf zum Einsatz kommt.

Strukturierte digitale Datenerfassung

Die Datenerfasser sollen bei Kundenterminen in Wohnhäusern in der Lage sein, mit der iOS-App strukturiert die Merkmale und Eigenschaften der Gebäudeteile digital zu erfassen. Dazu müssen die Nutzer mit der App zu jedem Gebäudeteil alle relevanten Daten textuell festhalten können.

Erfassung von gebäudespezifischen Besonderheiten

Da jedes Gebäude und Projekt unterschiedlich ist, muss die Software das Erfassen von gebäudespezifischen Besonderheiten ermöglichen. Der Nutzer muss in der Lage sein, vom Normalfall abweichende Merkmale effizient zu erfassen, um spezifische Eigenschaften und Anforderungen dokumentieren zu können.

Beliebige Reihenfolge der Datenerfassung

Da die Analyse gezeigt hat, dass die Reihenfolge, in der die Gebäudeteile beim Rundgang der Bestandsaufnahme erfasst werden, je nach Gebäude abweichen kann, muss die Software es erlauben, dass die Datenerfasser während des Rundgangs die Gebäudebauteile in einer zum Gebäude passenden Reihenfolge erfassen können. Diese Flexibilität gewährleistet, dass sich die App in den Arbeitsablauf der Nutzer integriert und diesen nicht strikt vorgibt.

Flexibles Navigieren während der Datenerfassung

Während des Rundgangs oder des Kundengesprächs kann es vorkommen, dass sprunghaft zwischen verschiedenen Aspekten des Gebäudes gewechselt wird. Die App muss es daher ermöglichen, dass der Nutzer schnell an die Stelle navigieren kann, an der diese Daten erfasst werden können, um den Arbeits- oder Gesprächsfluss nicht zu unterbrechen.

Fotodokumentation

Gewisse Eigenschaften von Gebäudeteilen lassen sich nur schwer als strukturierte textuelle Daten erfassen, weshalb Fotos, zum Beispiel von Heizungsanlagen oder Dachgauben, gemacht werden. Dies muss durch die App möglich sein, um die Fotos direkt in die Gebäudebestandsaufnahme zu integrieren.

Dokumentation der Kundenwünsche

Da der Mitarbeiter, der den Sanierungsfahrplan erstellt, nicht derselbe sein muss wie jener, der die Vor-Ort-Dokumentation durchgeführt hat, muss sichergestellt sein, dass neben den technischen Gebäudedaten auch die Wünsche und Bedürfnisse des Kunden im Kundengespräch vor Ort mit der App dokumentiert werden können. Darunter fällt beispielsweise, welche Sanierungsmaßnahmen geplant oder gewünscht sind und wie hoch das Budget dafür ist. Dies stellt sicher, dass der Projektbearbeiter alle relevanten Informationen in den Sanierungsfahrplan einbeziehen kann und die spezifischen Anforderungen der Kunden berücksichtigt werden. Somit wird das zuvor genannte Problem des Informationsverlusts bei Arbeitsteilung adressiert.

Nutzbarkeit in eingeschränkten Umgebungen

Die Analyse hat ergeben, dass die Mitarbeiter bei der Gebäudebestandsaufnahme häufig in unterschiedlichen und teilweise herausfordernden Umgebungen arbeiten müssen. Neben normalen Wohnräumen befinden sich die Mitarbeiter auch in Kellern, Dachböden oder anderen schwer zugänglichen Bereichen. Diese Orte sind häufig durch eingeschränkte Sichtverhältnisse und begrenzte Beweglichkeit gekennzeichnet.
Um die Nutzbarkeit der zukünftigen Software in diesen unterschiedlichen Umgebungen zu gewährleisten, müssen mehrere Aspekte berücksichtigt werden. Erstens sollte die Benutzeroberfläche der App so gestaltet sein, dass sie auch unter eingeschränkten Lichtverhältnissen ablesbar und bedienbar ist. Zweitens soll verhindert werden, dass Daten durch Missgeschicke oder ungewollte Touch-Gesten verloren gehen.

4.3.3 Projektbearbeiter

Für die Gruppe der Projektbearbeiter, deren Nutzungskontext sich stark von dem der Datenerfasser unterscheidet, ergeben sich aus der Analyse die folgenden Nutzungsanforderungen.

Bearbeitung am Desktop-PC

Da die Projektbearbeitung im Büro am Desktop-PC stattfindet, muss die Software auch auf dieser Plattform nutzbar sein. Dies gewährleistet, dass die beim Vor-Ort-Termin am iPad erfassten Daten nahtlos weiterverarbeitet werden und die Mitarbeiter ihre gewohnten Arbeitsumgebungen beibehalten können.

Kompatibilität der Formulare zur Energieberater-Software

Um den Prozess der Datenübertragung in die Energieberater-Software zur Erstellung des iSFPs effizient zu gestalten, sollen die Formulare zur Datenerfassung einen ähnlichen Aufbau wie die der Energieberater-Software haben. Dies gewährleistet eine reibungslose Integration und minimiert den Aufwand für Datenkonvertierungen, sodass die Erstellung des Sanierungsfahrplans effizienter erfolgen kann.

Abdeckung aller nötigen Angaben für die Projektbearbeitung

Damit der Projektbearbeiter den iSFP ohne detaillierte Rückfragen an den Datenerfasser erarbeiten kann, müssen bei der Datenerfassung durch die App alle Angaben abgefragt und abgedeckt werden, die für die Erstellung des iSFPs notwendig sind.

Verantwortungszuordnung der Datenerfassung

Da der Projektbearbeiter unter Umständen die Datenerfassung nicht selbst durchgeführt hat, muss es ihm möglich sein, zu sehen, welcher Mitarbeiter die Daten erfasst und geändert hat. Dies ist insbesondere dann notwendig, wenn der Projektbearbeiter Rückfragen zu den erfassten Daten an den Datenerfasser hat. Durch die Nachverfolgbarkeit von Änderungen und die Zuordnung zu den jeweiligen Verantwortlichen wird eine effiziente Zusammenarbeit innerhalb des Teams sichergestellt.

5

Konzept

Auf Basis des analysierten Nutzungskontextes und der definierten Nutzungsanforderungen wird im Folgenden ein konzeptioneller Aufbau für die Benutzungsschnittstelle der Software entworfen.
Das Konzept wird in Form von Wireframes für die einzelnen Seiten und Funktionen erstellt, die als Orientierung bei der Implementierung dienen. Die Wireframes sind bewusst in Graustufen und einfach gehalten, da in dieser Phase zunächst die Funktionen und deren grundlegende Struktur innerhalb der Benutzungsschnittstelle im Vordergrund stehen.

5.1 Authentifizierung und Autorisierung

Damit die App ausschließlich für Mitarbeiter der Firma TRAGWERK nutzbar ist und so die Kundendaten geschützt werden, soll die Software über Benutzerkonten für jeden Mitarbeiter verfügen. Beim Öffnen der App wird eine Login-Seite angezeigt, auf der sich der Nutzer einloggen oder die Änderung seines Passworts anfordern kann. Bei erfolgreicher Authentifizierung gelangt der Nutzer zur Startseite der App.
Zusätzlich zur Authentifizierung werden dem Nutzer nur Funktionen angezeigt, für die er autorisiert ist. Die Benutzerkonten verfügen dazu über unterschiedliche Rollen, entweder „interner Mitarbeiter“ oder „Administrator“. Administratoren können im Gegensatz zu internen Mitarbeitern auch Benutzerkonten erstellen oder löschen.

5.2 Basisseiten

Grundsätzlich wird die App über eine Reihe von Seiten für die Grundfunktionen verfügen, die der App eine grundlegende Struktur geben. Diese werden im Folgenden als Basisseiten bezeichnet.
Bei allen Basisseiten befindet an der linken Seite ein Seitenmenü, das etwa 20 Prozent der Breite einnimmt und auf jeder Basisseite konstant sichtbar bleibt. Dieses Seitenmenü ist in drei Teile gegliedert: oben befindet sich das Firmenlogo, in der Mitte eine Liste aller Basisseiten als Buttons mit der farblichen Hervorhebung der

aktuell geöffneten Seite und unten ein Button zur Benutzerkonto-Seite. Das Seitenmenü ermöglicht eine einfache Navigation zwischen den Basisseiten und bietet dem Nutzer eine Orientierungshilfe.
Der rechte Teil des Bildschirms, der etwa 80 Prozent der Breite einnimmt, ist für die aktuell gewählte Basisseite reserviert.

5.2.1 Startseite

Die Startseite ist der Einstiegspunkt in die App, ermöglicht einen schnellen Zugriff auf häufig verwendete Funktionen und zeigt bei Bedarf Hinweise zum Systemzustand an.
Diese verfügt, wie in Abbildung 5.1 zu sehen ist, über eine Übersicht der zuletzt geöffneten Projekte des Nutzers in Form von mehreren Kacheln mit den Grunddaten des Projekts. Diese Kacheln dienen als Schnellzugriff zu den zuletzt bearbeiteten Projekten. Sie sind dafür gedacht, dass die Datenerfasser, die ein Projekt vor einem Kundentermin zur Bestandsaufnahme am PC erstellen, dieses beim Termin direkt auf dem iPad über die Startseite öffnen können. Dadurch muss das Projekt nicht mehr in der Projektverwaltung gesucht werden.
Sofern es lokale Änderungen an Daten gibt, die bisher nicht mit der Cloud-Datenbank synchronisiert wurden, wird auf der Startseite ein kleiner Hinweis in Form eines Banners angezeigt. Ein Klick auf das Banner führt zu einer Seite, die den Synchronisierungsstatus in Form einer Liste aller bisher nicht synchronisierten Änderungen zeigt. Dies hilft den Datenerfassern nach einer Bestandsaufnahme nachzuvollziehen, welche Änderungen bisher nicht in der Cloud-Datenbank gespeichert wurden.

5.2.2 Kundenverwaltung

Die Kundenverwaltungsseite, deren Wireframe in Abbildung 5.2 zu sehen ist, ermöglicht es Benutzern, Kunden zu verwalten. Die Kunden werden dabei in einer Liste von Kacheln angezeigt, die als Titel den Kundennamen und als Untertitel den Ort und den Kundentyp anzeigen. Durch einen Klick auf diese Kacheln gelangt der Nutzer zur Kundendetailseite. Um einen spezifischen Kunden zu suchen, gibt es oberhalb der Liste die Möglichkeit, die Kunden nach Kundentyp zu filtern und nach Namen zu suchen.
In der rechten unteren Bildschirmecke befindet sich zudem ein Aktionsbutton. Dieser ist fixiert, damit er auch beim Scrollen durch die Liste immer sichtbar bleibt. Per Klick navigiert der Nutzer zu einer Seite mit einem Formular zur Erstellung des Kunden, welches das Gleiche ist, das auch auf der Kundendetailseite zum Anzeigen der Kundendetails verwendet wird.

5.2.3 Projektverwaltung

Die Projektverwaltungsseite hat das gleiche Layout wie die Seite zur Verwaltung der Kunden und ist in Abbildung 5.3 skizziert. Im Unterschied zur Kundenverwaltung können die Projekte nach Projektstatus gefiltert und nach Kundenname

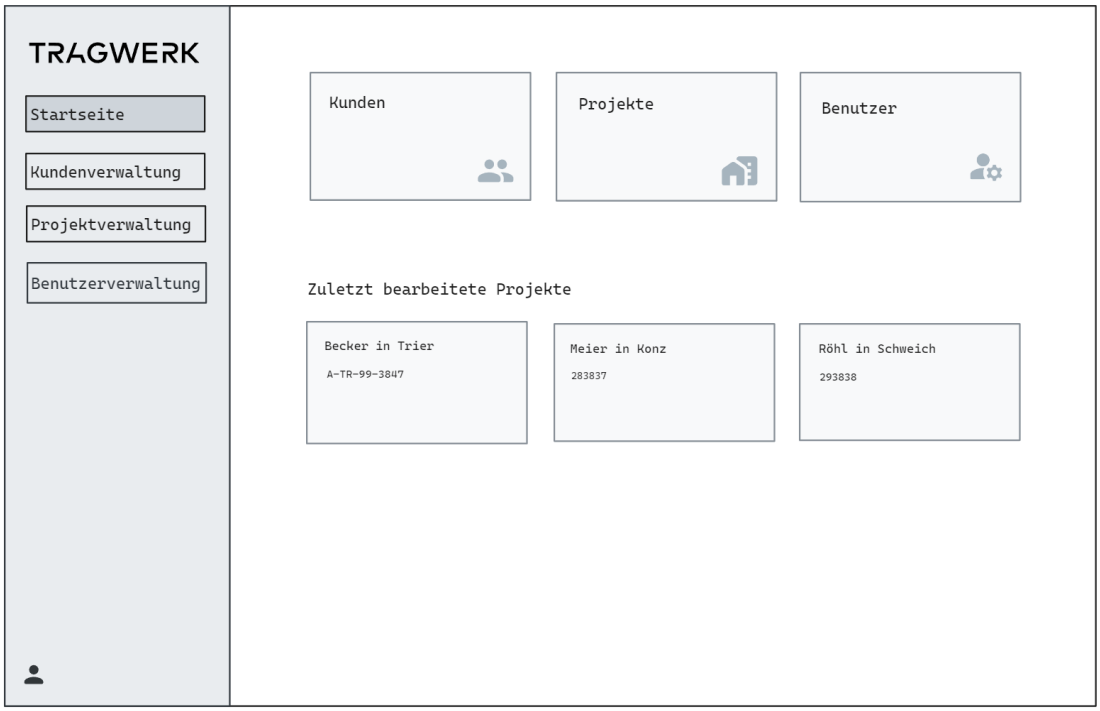


Abbildung 5.1: Wireframe der Startseite inklusive Seitenmenü

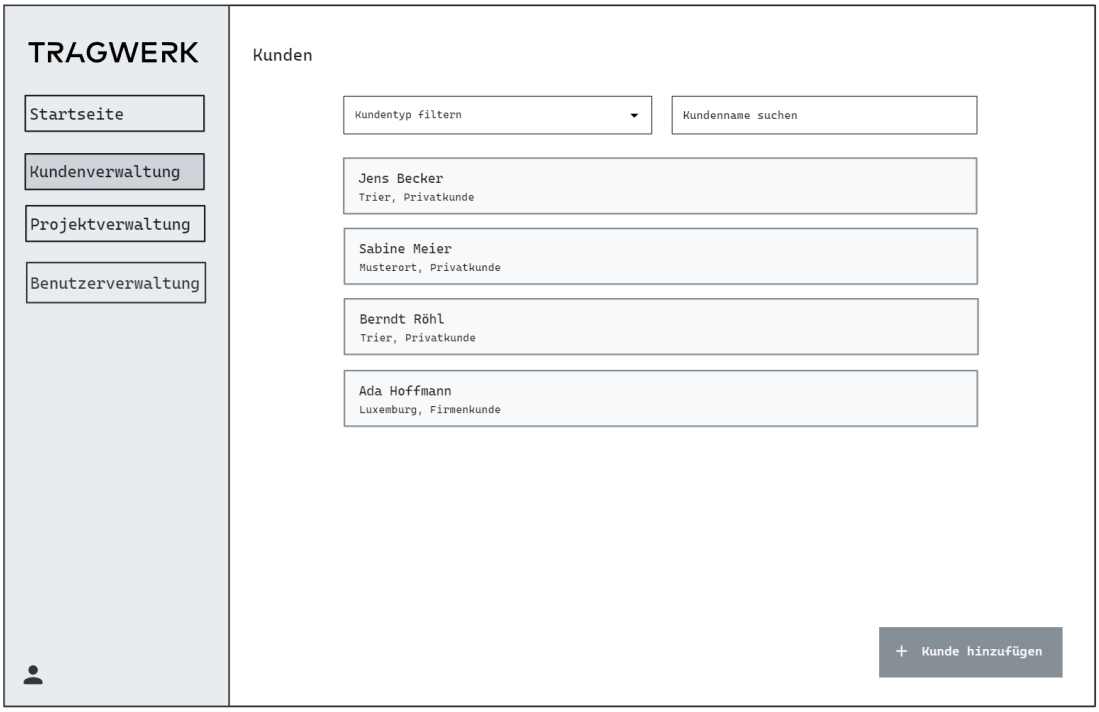


Abbildung 5.2: Wireframe der Kundenverwaltung

oder Projektnummer durchsucht werden. Die Kacheln zeigen den Projekttitel an, der sich beim Praxispartner immer aus dem Nachnamen des Kunden und der Projektadresse zusammensetzt.
Ebenso befindet sich in der unteren rechten Ecke des Bildschirms ein Button zur Erstellung eines Projekts. Dieser führt jedoch nicht direkt zur Seite zum Erstellen eines Formulars, sondern zunächst zur Kundenverwaltung mit einem Hinweisbanner, dass zunächst ein Kunde ausgewählt werden muss, bevor zu dem Kunden ein Projekt erstellt werden kann.

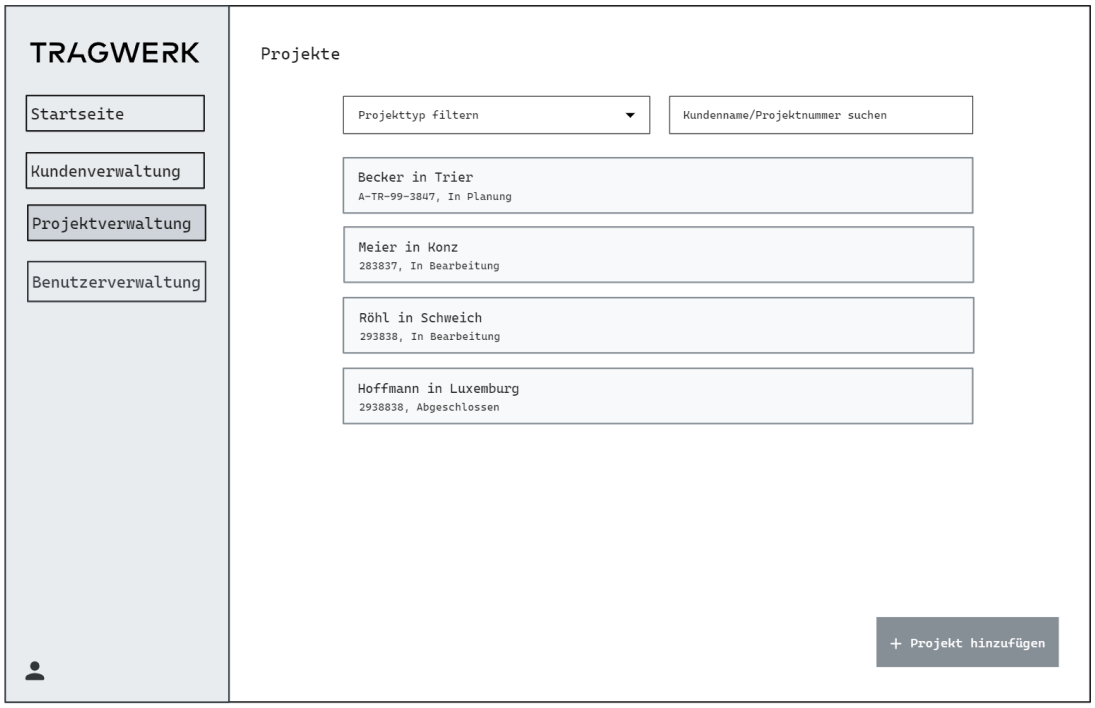


Abbildung 5.3: Wireframe der Projektverwaltung

5.2.4 Weitere Seiten

Neben den zuvor gezeigten Seiten wurden noch weitere Basisseiten konzipiert, die jedoch für diese Arbeit nebensächlich sind und daher nicht weiter erläutert werden.
Dazu gehört die Benutzerverwaltungsseite, die nur für Benutzer mit der Rolle „Administrator“ sichtbar ist und im Aufbau der Projekt- und Kundenverwaltung ähnelt. Zudem gibt es eine Benutzerseite, die es dem Nutzer ermöglicht, seine Benutzerkontodetails wie E-Mail und Name einzusehen, das Passwort zu ändern oder sich auszuloggen. Von dieser Seite gelangt der Nutzer auch auf die App-Informationsseite, welche Informationen zur App wie die Version, das Impressum und die Datenschutzerklärung anzeigt. Über diese Seite können die Nutzer bei Bedarf auch Kontakt zum Entwickler herstellen.

5.3 Detailseiten

Zu den Projekten und Kunden gibt es jeweils Detailseiten, die aufgerufen werden können, wenn auf einer Basisseite ein Projekt oder Kunde ausgewählt wird. Diese Detailseiten bieten eine umfassende Übersicht über alle Informationen zu dem jeweiligen Projekt oder Kunden und ermöglichen es den Benutzern, alle relevanten Daten an einem Ort zu finden.
Im Gegensatz zu den Basisseiten erscheint bei den Detailseiten in der oberen Leiste links neben dem Seitentitel ein Zurück-Button, um zu der jeweiligen vorherigen Basisseite zurückzukehren.

5.3.1 Kundenseite

Die Kundenseite zeigt, wie in Abbildung 5.4 skizziert, in der oberen Hälfte ein Formular mit den Kundendetails an. Über dieses können alle Kundendaten eingesehen und bearbeitet werden. Darunter wird, wie auch bei allen anderen Formularen innerhalb der App, angezeigt, welcher Mitarbeiter den Datensatz zu welchem Zeitpunkt erstellt und zuletzt bearbeitet hat.
Unterhalb dieses Formulars wird in einer Liste aller dem Kunden zugeordneten Projekte angezeigt. Hier werden die gleichen Kacheln wie bei der Projektverwaltungsseite wiederverwendet. Ebenso gibt es einen Aktionsbutton zum Erstellen eines Projekts für den Kunden.

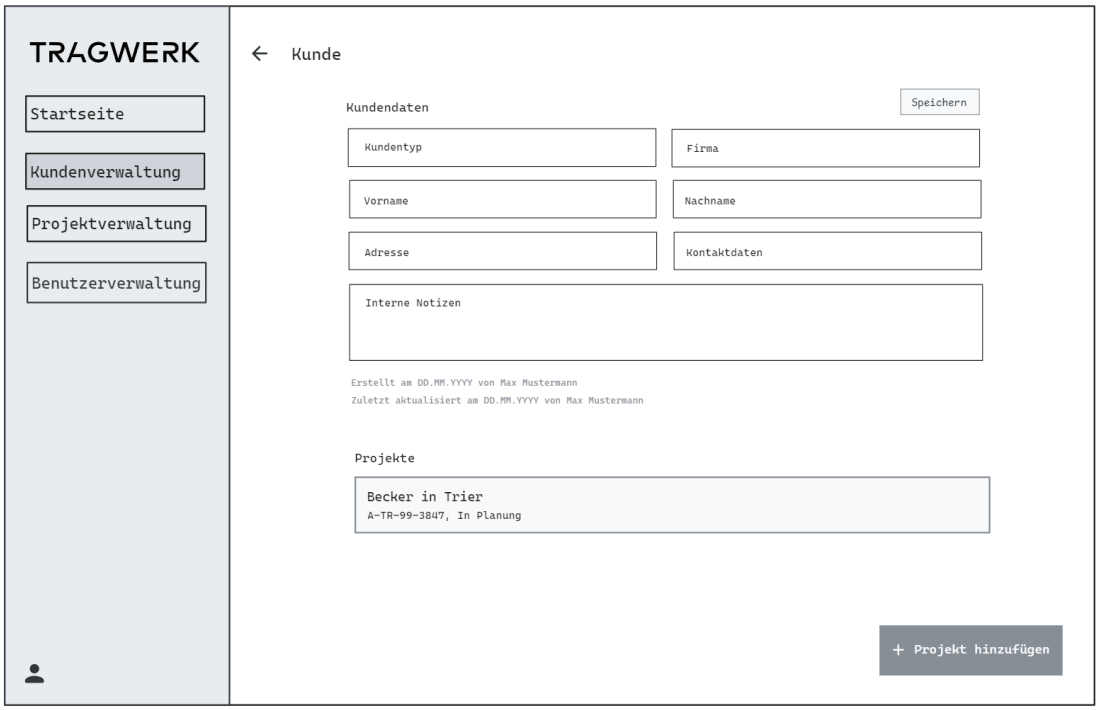


Abbildung 5.4: Wireframe der Kundendetailseite

5.3.2 Projektseite

Auf der Projektdetailseite hat der Nutzer die Möglichkeit, die Projektdetails einzusehen und über einen Button zu bearbeiten. Zusätzlich sieht er die Details des Kunden, dem das Projekt zugeordnet ist. Diese Seite ist in Abbildung 5.5 zu sehen. Unterhalb dieser Daten gibt es zwei Abschnitte, über die der Nutzer zu den Funktionen zur Erfassung der Kundenanforderungen und der technischen Gebäudebestandsaufnahme für das Projekt gelangt. Diese werden in den folgenden Abschnitten genauer erklärt.

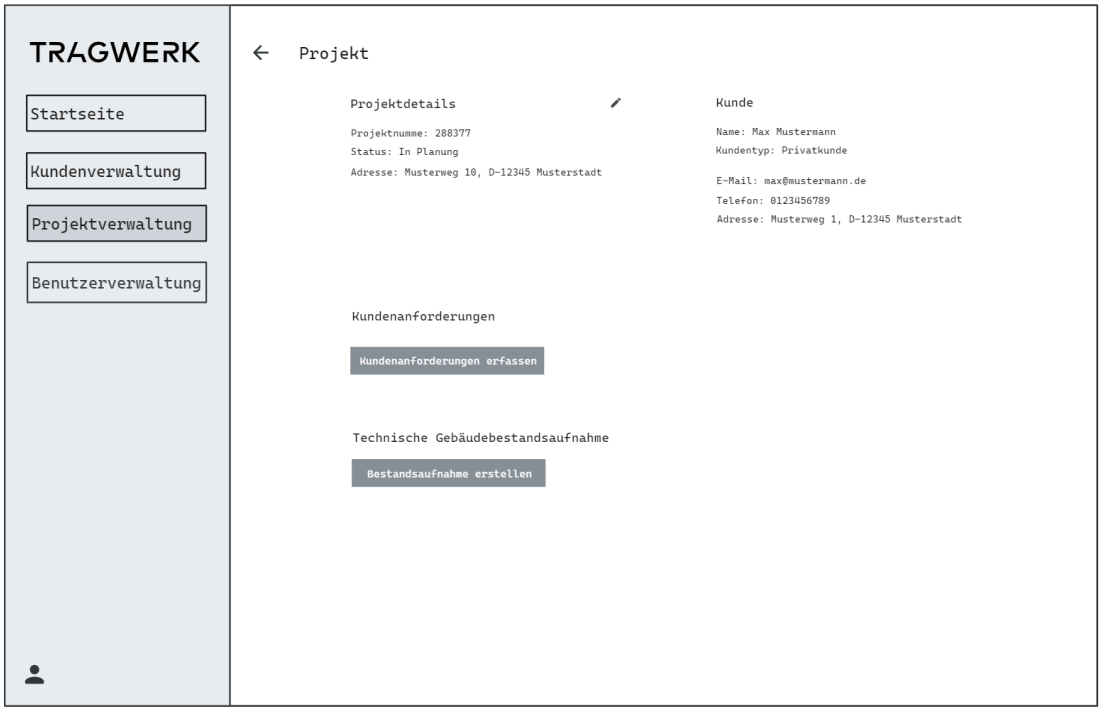


Abbildung 5.5: Wireframe der Projektdetailseite

Gebäudebestandsaufnahme

Die Kernfunktion der App besteht darin, technische Gebäudebestandsaufnahmen zu erstellen. Damit die Merkmale der Bauteile strukturiert erfasst werden können, werden Formulare verwendet. Die Übersichtlichkeit wird gewährleistet, indem es pro Bauteil eine eigene Seite mit einem vordefinierten Formular festen Schemas gibt.
Als Grundlage für Formulare und dessen Schema, also die Felder, die innerhalb der Formulare ausgefüllt werden können, dient die aktuelle Vor-Ort-PDF-Checkliste der Firma, welche in Zusammenarbeit mit dem Praxispartner um weitere Angaben ergänzt wurde.
Die Analyse hat gezeigt, dass es bei einem Vor-Ort-Termin wichtig ist, schnell zwischen den verschiedenen Formularen zu wechseln und insbesondere während eines

Gesprächs die Kategorien nicht immer chronologisch abgearbeitet werden können. Daher wird am linken Bildschirmrand eine Seitenleiste, ähnlich der Seitenleiste der Basisseiten, zu sehen sein, die eine Liste aller Formulare anzeigt und es ermöglicht, schnell zwischen den verschiedenen Formularen zu navigieren und diese in einer beliebigen Reihenfolge abzuarbeiten.
Um während der Bestandsaufnahme vor Ort Fotos zu einem Bauteil hinzuzufügen, gibt es die Möglichkeit, pro Formularseite einen Fotobereich zu öffnen. Dazu gibt es am rechten oberen Bildschirmrand eine Funktion zum Wechseln zwischen Formular und Fotodokumentation.
Exemplarisch ist ein Wireframe zum Fassadenformular der Bestandsaufnahme in Abbildung 5.6 dargestellt.

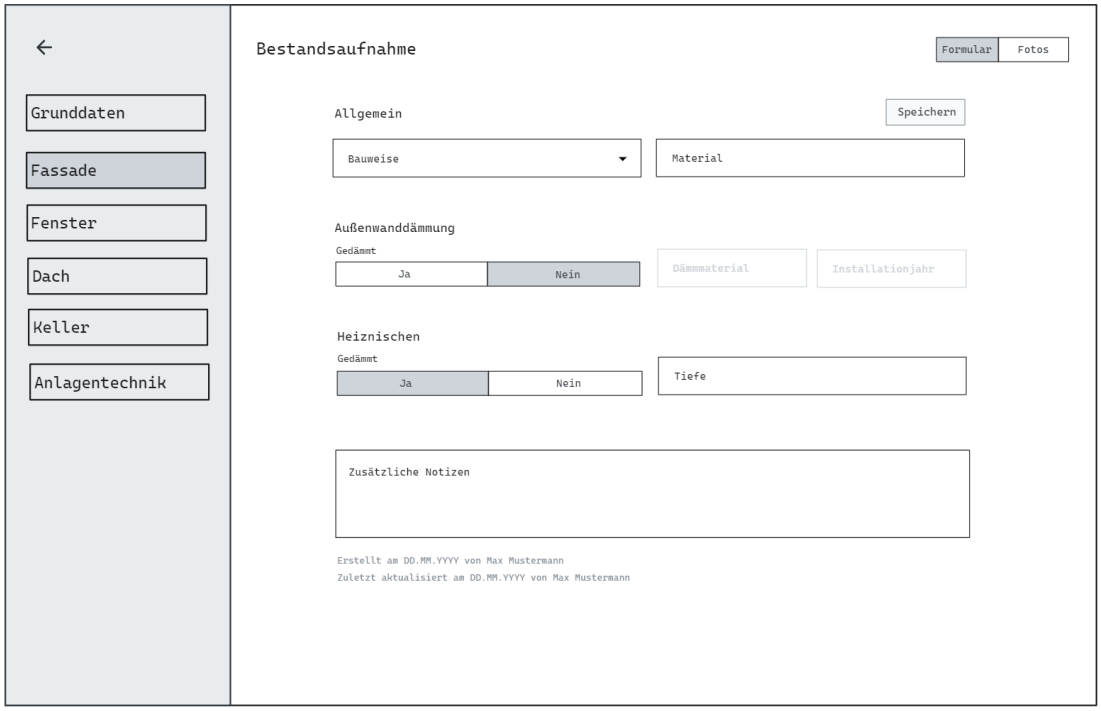


Abbildung 5.6: Wireframe des Fassaden-Formulars der Bestandsaufnahmenfunktion

Formulare

Im Formular-Tab werden die Formularfelder thematisch in Abschnitte zusammengefasst. Zum Beispiel hat die Kategorie „Allgemein“ des Formulars zur Fassade die Felder „Bauweise“ und „Material“. Dabei ist das Feld zur Bauweise der Fassade ein Dropdown-Auswahlfeld mit den Optionen „Monolithisches Mauerwerk“, „Zweischaliges Mauerwerk“, „Holz“ und „Beton“. Diese Felder werden überall dort verwendet, wo es eine gängige Auswahl an Optionen gibt und ermöglichen somit eine

schnellere Eingabe als Textfelder. Zudem führen diese zu einheitlicheren Angaben und erleichtern so die Projektbearbeitung. Jedoch können Auswahlfelder nicht dort verwendet werden, wo es eine große Varianz an Eingabemöglichkeiten gibt, beispielsweise beim Material der Fassade, dort werden stattdessen Freitextfelder verwendet.
Neben diesen beiden Feldtypen gibt es auch „Ja“/„Nein“-Felder, wie in Abbildung 5.6 bei der Außenwanddämmung zu sehen. Hierbei ist eine Besonderheit der Formulare, nämlich bedingte Felder, zu sehen. Nach Gesprächen mit dem Praxispartner kommt es oft vor, dass bestimmte Daten nur dann erfasst werden können und müssen, wenn zuvor eine gewisse Eingabe getätigt wurde. Beispielsweise kann das Material und das Installationsjahr der Fassadendämmung nur dann eingetragen werden, wenn überhaupt eine Fassadendämmung vorhanden ist. Diese bedingten Felder werden so lange deaktiviert und ausgegraut, bis die entsprechende Bedingung für den Wert eines anderen Feldes erfüllt ist. Dies erhöht zudem die Übersichtlichkeit der Formulare.
Darüber hinaus gibt es noch weitere Arten von Formularfeldern wie Ganzzahl- und Dezimalzahlfelder.
Die Formularfelder sind dabei alle optional, da jede Bestandsaufnahme unterschiedlich ist und manche Felder nicht für jedes Gebäude relevant sind oder erst im Nachhinein ergänzt werden können.
Unter den Formulargruppen befindet sich ein Freitextfeld für zusätzliche Notizen und Informationen zu den erfassten Daten. Dieses Feld adressiert die Nutzungsanforderung der Erfassung von gebäudespezifischen Besonderheiten. Dies stellt sicher, dass alle relevanten Informationen erfasst werden können, selbst wenn unerwartete oder spezielle Besonderheiten auftreten. Dort können aber auch weitere Angaben vermerkt werden, die nicht vom Formular abgedeckt werden, beispielsweise dass das Dach eines Gebäudes demnächst neu gedeckt und gedämmt wird, was bei der späteren Weiterbearbeitung des Projekts berücksichtigt werden muss.
Durch diese Struktur wird die Erfassung der Gebäudedaten systematisch und effizient gestaltet, während gleichzeitig ausreichend Flexibilität für individuelle Anmerkungen und besondere Fälle gewährleistet ist.
Da es den Projektbearbeitern möglich sein muss, zu erkennen, wer die Daten vor Ort erfasst hat, wird dies unterhalb jedes Formulars in Form des Namens des Bearbeiters und dem Datum der Erstellung und letzten Bearbeitung angezeigt.
Um die Erkenntnis aus der Analyse zu berücksichtigen, dass Mitarbeiter während der Bestandsaufnahme vorwiegend den Apple Pencil anstelle der Tastatur verwenden, da dies auf größeren Geräten wie iPads effizienter ist, sollen die Formularfelder eine Handschrifterkennung unterstützen. Dadurch ist es dem Benutzer möglich, Text oder Zahlen handschriftlich einzugeben, die dann direkt erkannt und umgewandelt werden.
Um die Daten eines Formulars zu speichern, wird oberhalb des Formulars ein Speichern-Button hinzugefügt, der automatisch aufleuchtet, sobald eine Änderung an den Formulardaten vorgenommen wurde.
Wie in den Anforderungen aufgenommen, soll verhindert werden, dass die Bestandsaufnahmeseite ungewollt geschlossen und damit alle ungespeicherten Än-

derungen verloren gehen. Deshalb erscheint in diesen Fällen ein Pop-up-Fenster, das den Nutzer auffordert, zunächst die Formulare mit Änderungen zu speichern, bevor er zurücknavigieren kann.

Duplizierbare Formulargruppen

Eine Besonderheit bei den Formularen zur Gebäudebestandsaufnahme stellt das Formular zu den Fenstern dar. Da ein Gebäude, besonders im Bestandsbau, oft mehrere verschiedene Fenstertypen hat und die Merkmale dieser relevant für den iSFP sind, reicht hier ein Formular zur Angabe der Fenstereigenschaften nicht aus. Stattdessen muss es je nach Gebäude beliebig viele dieser Formulare geben.
Aus diesem Grund gibt es auf der Formularseite zu den Fenstern die Möglichkeit, mehrere Fenstertyp-Formulargruppen zu erstellen und zu löschen, welche in sich ein festes Schema haben. Dies ist in Abbildung 5.7 zu erkennen.
Unterhalb dieser Liste an Formulargruppen ist wie auch auf den anderen Seiten ein Freitextfeld für zusätzliche Notizen abgebildet.

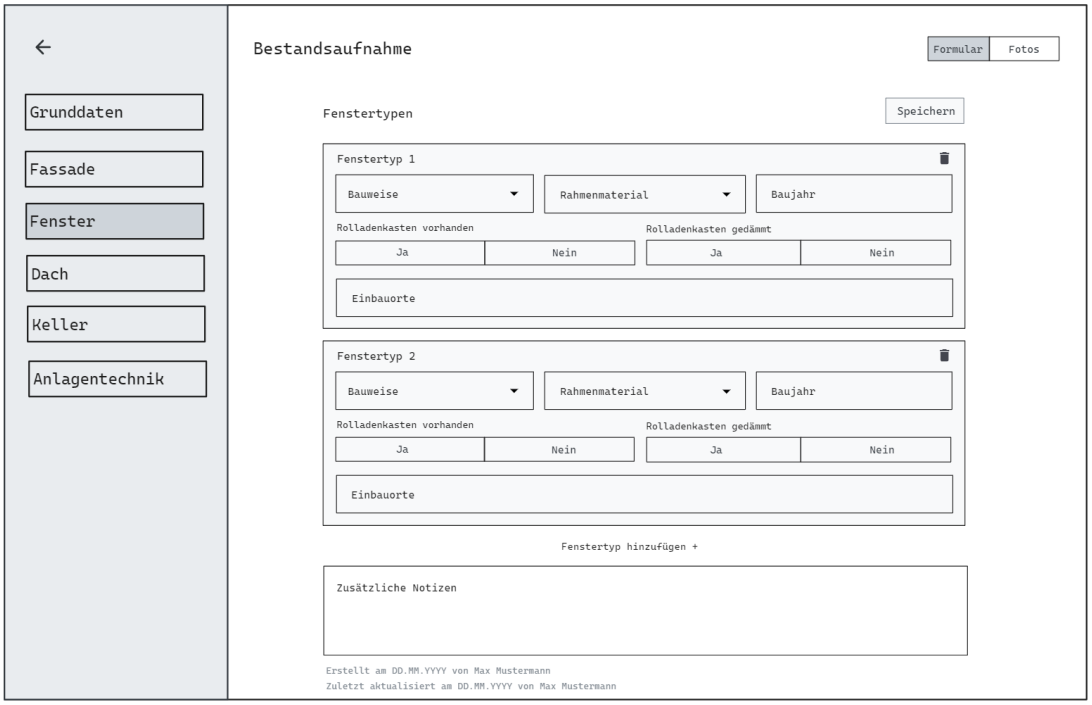


Abbildung 5.7: Wireframe des Fenster-Formulars der Bestandsaufnahmenfunktion

Fotodokumentation

Der zweite Tab, der als die Rückseite eines Formulars betrachtet werden kann und dessen Wireframe in Abbildung 5.8 zu sehen ist, enthält die durch den Datenerfasser vor Ort aufgenommenen Fotos zum jeweiligen Bauteil. Jedes Foto ist somit

einem Bauteil der Bestandsaufnahme zugeordnet.
Beim Hinzufügen eines Fotos per Kamera kann zudem eine optionale Beschreibung bzw. Notiz zum Fotos hinzugefügt werden. Diese Notiz kann nützliche Kontextinformationen für den Projektbearbeiter enthalten. Beispielsweise könnte ein Foto von einem Balkon die Information enthalten, dass dieser im Rahmen einer Umbaumaßnahme abgerissen werden soll.

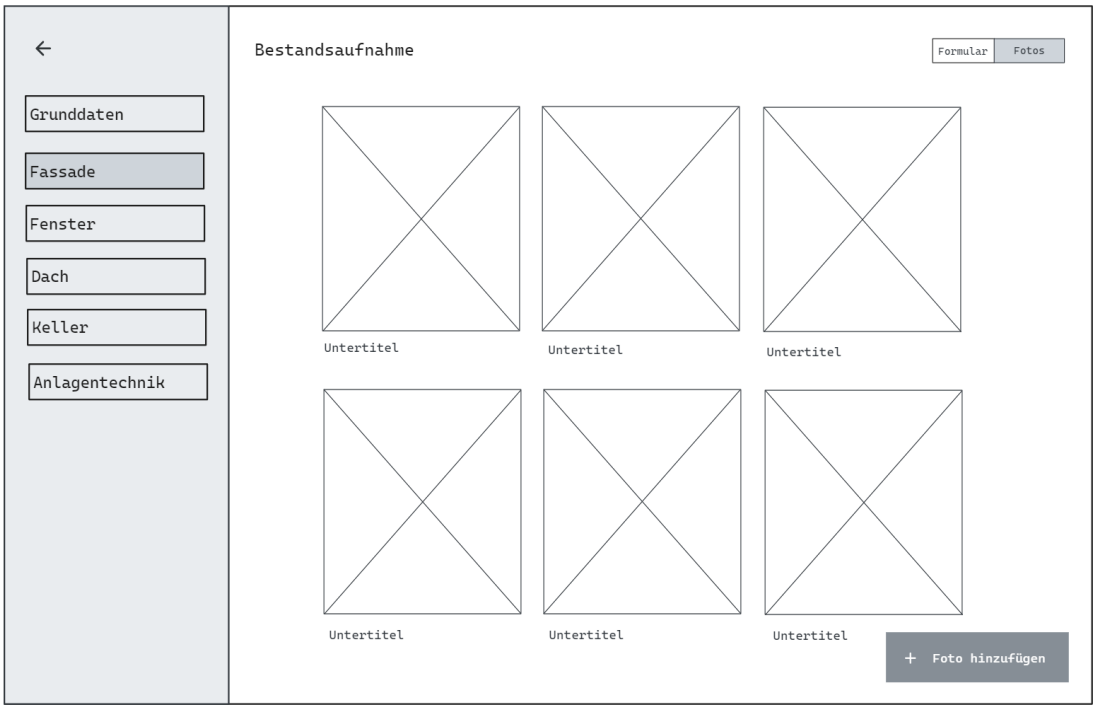


Abbildung 5.8: Wireframe der Fassaden-Fotodokumentation der Bestandsaufnahmenfunktion

Kundenanforderungen

Die Funktion zur Erfassung der Kundenanforderungen ist ähnlich wie die Gebäudebestandsaufnahme aufgebaut. Sie verfügt ebenso über eine Seitenleiste und Formulare zur Datenerfassung, mit dem Unterschied, dass hier keine Fotodokumentation notwendig ist.
Vom Praxispartner wurden hier drei Formulare vorgegeben: eins zum Erfassen der Details zu den Bewohnern des Hauses, eins zu den Kundenwünschen und eins zum Budget und Zeitplan des Projekts.
Eine Besonderheit dieser Formulare ist, dass es zum einen Währungsfelder, beispielsweise zur Angabe des Budgets, gibt und auch Felder, bei denen Mehrfachauswahlen möglich sind. Ein Beispiel hierzu ist das Formularfeld zur Auswahl der notwendigen Investitionen, wie in Abbildung 5.9 zu erkennen ist.

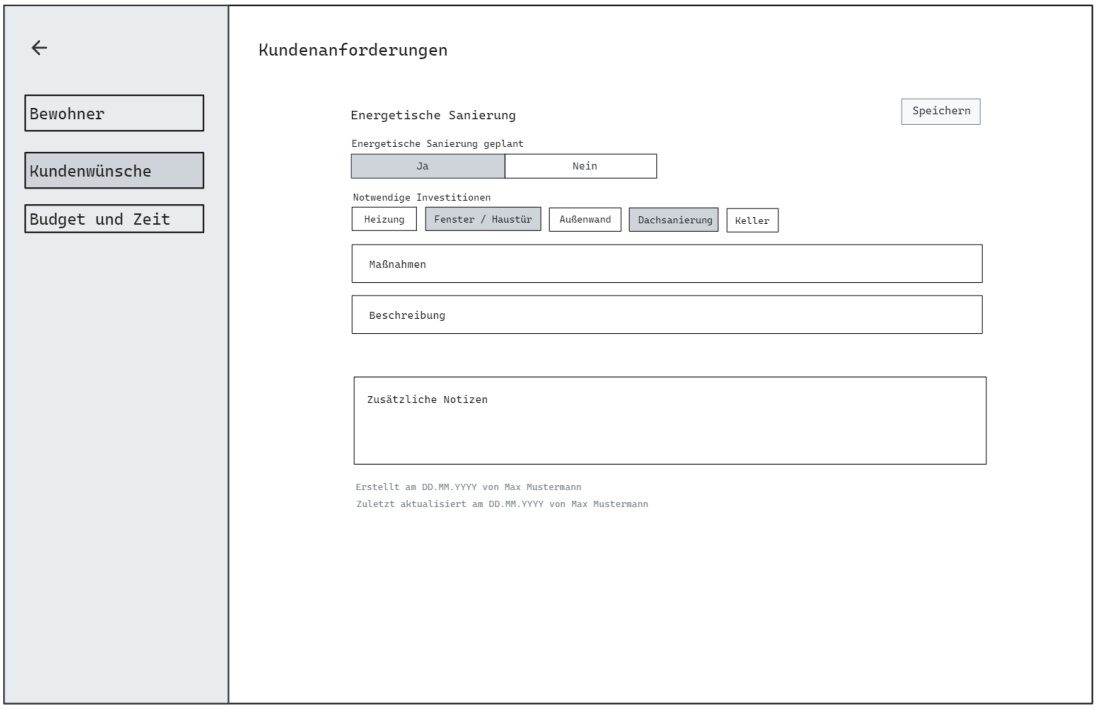


Abbildung 5.9: Wireframe des Formulars zur Erfassung der Kundenwünsche im Rahmen der Kundenanforderungsfunktion

5.3.3 Offlinefähigkeit und Synchronisierung

Aus der Funktionsanforderung der Offlinefähigkeit ergeben sich einige weitere zu konzipierende Funktionen. Dazu zählt zum einen, dass beim Öffnen eines Formulars dieses zunächst lokal geladen und dann, falls möglich, von der Cloud-Datenbank abgerufen werden soll. Während dieses Ladevorgangs wird neben dem Speichern-Button eine Ladeanimation mit dem Hinweistext „Synchronisieren“ angezeigt, um dem Nutzer zu signalisieren, dass versucht wird, die neuesten Daten des Formulars von der Cloud-Datenbank zu laden. Gelingt dies nicht, wird stattdessen ein Icon mit Hinweis angezeigt, dass der Nutzer offline ist und das Formular daher möglicherweise veraltete Daten anzeigt.
Da Änderungen an den Kunden- und Projektdaten während einer Vor-Ort Bestandsaufnahme ohne eine Internetverbindung, also offline, durchgeführt werden, muss die App diese lokal zwischenspeichern und kann diese erst zu einem späteren Zeitpunkt an die Cloud-Datenbank senden. Um dem Nutzer eine Rückmeldung darüber zu geben, welche Änderungen bereits mit der Cloud-Datenbank synchronisiert wurden, kann dieser über einen Button in einem Hinweisbanner auf der Startseite auf eine Seite gelangen, die eine Liste aller ausstehenden unsynchronisierten Datenänderungen anzeigt. Sobald alle Änderungen synchronisiert wurden, wird dies dem Nutzer auf der Seite angezeigt und das Banner auf der Startseite wird ausgeblendet.

5.4 Corporate Design

Das Design der Benutzungsschnittstelle der App wird dem Corporate Design der Firma TRAGWERK entsprechen, welches eine spezifische Schriftart, das Firmenlogo und eine festgelegte Farbpalette umfasst.
Die Farbpalette besteht aus einem leuchtenden Blau, einem Graublau, Hellgrau, Schwarz und Weiß. Dieses Farbschema wird bei der Umsetzung des UI-Designs verwendet, um eine seriöse und ruhige Wirkung zu erzielen. Ebenso wird eine quadratische Variante des Firmenlogos für das App-Icon verwendet.
Grundsätzlich wird das Design der App hell und schlicht gehalten, um die Nutzungsanforderung der Nutzbarkeit in eingeschränkten Umgebungen, wie schlechten Lichtbedingungen, zu adressieren.

6

Implementierung

Auf Basis des erstellten Konzepts wird das Softwaresystem in Form einer iOS und Web-App mit den genannten Anforderungen und Funktionen implementiert. Aus Zeitgründen musste die Implementierung der Fotodokumentation im Rahmen dieser Arbeit ausgelassen werden, alle anderen konzipierten Funktionen wurden umgesetzt, wobei ein besonderer Fokus auf die Bestandsaufnahme-Formulare gelegt wurde. Dieses Kapitel beschreibt die technischen Implementierungsdetails zur Umsetzung.

6.1 Grundkomponenten der Softwarearchitektur

Die Softwarearchitektur der entwickelten App besteht aus mehreren grundlegenden technischen Komponenten, die in Abbildung 6.1 inklusive der Datenflussrichtungen zwischen diesen gezeigt werden. Im Folgenden werden diese Komponenten kurz beschrieben und die verwendeten Technologien sowie die Gründe für deren Auswahl erläutert.

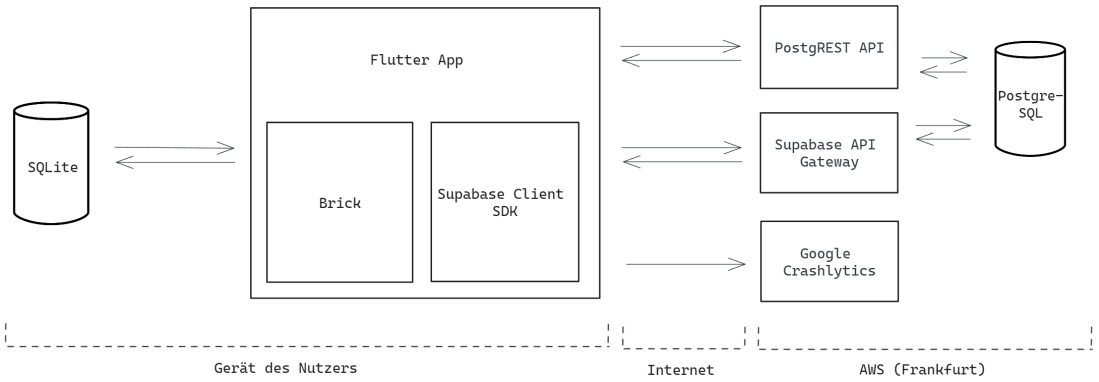


Abbildung 6.1: Blockdiagramm der Softwarearchitektur auf einem hohen Abstraktionsniveau inklusive Datenflussrichtungen

6.1.1 Flutter

Die App wird mit dem quelloffenen UI-Entwicklungskit namens Flutter von Google entwickelt.
Mit Flutter können plattformübergreifende Apps aus einer Codebasis in der Programmiersprache Dart erstellt werden. Flutter kompiliert zu ARM- und Intel-Maschinencode sowie zu JavaScript, um eine hohe Performance auf allen Plattformen zu erzielen. Eine weitere Charakteristik von Flutter ist die hohe Produktivität bei der Entwicklung sowie die flexible Darstellung von Inhalten, da Flutter die vollständige Kontrolle über jedes Pixel der App ermöglicht [Gooa].
Ein wesentlicher Grund für die Entscheidung, Flutter zu verwenden, ist die Notwendigkeit, eine App zu entwickeln, die sowohl auf iOS als auch im Web funktioniert. Es wäre zeitlich zu aufwändig, eine separate native iOS und Web-App zu erstellen. Zudem schließt der Auftraggeber nicht aus, die App in Zukunft auch um Funktionen für Kunden zu erweitern, weshalb auch Android unterstützt werden müsste. Flutter ermöglicht durch seine plattformübergreifende Entwicklung eine erhebliche Zeitersparnis.
Ein weiterer Grund sind die schnellen Entwicklungsfortschritte, die mit Flutter möglich sind. Flutter bietet beispielsweise eine Funktion namens Hot-Reload, die es ermöglicht, Änderungen am Code sofort in der laufenden App zu sehen, ohne sie neu starten zu müssen.
Des Weiteren ist der Quellcode öffentlich zugänglich, wodurch die Transparenz und Sicherheit erhöht wird. Die große Entwickler-Community stellt zudem zahlreiche Erweiterungen (im folgenden Packages genannt) zur Verfügung, die die Entwicklung erheblich erleichtern.

6.1.2 SQLite Datenbank

Um den Anforderungen einer offline fähigen App gerecht zu werden, ist die Verwendung einer lokalen Datenbank zur Speicherung aller App-Daten notwendig. Hierfür wird SQLite verwendet, welches eine „kleine, schnelle, in sich geschlossene, hochzuverlässige und mit vollem Funktionsumfang“ ausgestattete SQL (Structured Query Language) Datenbank ist. Diese ist in allen Smartphones integriert und der Quellcode ist öffentlich zugänglich. [SQL]
Die Wahl für SQLite hat den Grund, dass es weit verbreitet ist und es eine bekannte Flutter-Implementierung namens `sqflite` gibt, die eine gute Plattformabdeckung hat und daher auch Flutter-Web unterstützt, was eine Anforderung bei der Umsetzung der App ist.

6.1.3 Brick

Als grundlegende lokale Datenspeicher- und Synchronisierungsverwaltung kommt das Dart Package Brick zum Einsatz, welches die Verwaltung von Geschäftsdaten in der Anwendung ermöglicht, unabhängig davon, woher die Daten stammen. Brick wurde entwickelt, um sicherzustellen, dass Anwendungen offline funktionieren, indem die Geschäftsdaten in einer lokalen Datenbank gespeichert und bei Bedarf

mit externen Quellen synchronisiert werden. [Cou]
Brick wurde aufgrund seiner Offline-First-Abstraktionsebene zwischen GUI und der Datenbank ausgewählt und verwendet. Besonders überzeugt hat Brick durch seine Erweiterbarkeit, modulare Struktur sowie die Unterstützung verschiedener externer Datenquellen. Diese Eigenschaften machen Brick zu einer idealen Wahl für die Umsetzung dieser App.

Während Brick komplett clientseitig, also innerhalb der Flutter App, arbeitet, gibt es auch die Möglichkeit, Daten mithilfe eines externen Synchronisierungsservices zu replizieren und zu synchronisieren. Dazu ist das Hosten eines speziellen Servers notwendig, der dann als Zwischeninstanz zwischen App und Datenbank agiert. Zwei bekannte Anbieter für eine solche Lösung mit einer relationale Datenbank wie PostgreSQL sind PowerSync und ElectricSQL. Bei PowerSync ist dieser Dienst kostenpflichtig, kann aber wie bei ElectricSQL auch selbst gehostet werden. Da ElectricSQL noch keine offizielle Integration mit Dart bzw. Flutter anbietet, wäre nur die Benutzung von PowerSync in Frage gekommen.
Für diese Arbeit wurde sich jedoch gegen die Verwendung eines externen Synchronisierungsservices entschieden. Durch die Nutzung eines solchen Services entsteht eine große Abhängigkeit zu einem externen Anbieter, weil die Synchronisierung eine zentrale Komponente im Datenaustausch und damit der Funktionalität der App ist. Zusätzlich entstehen durch die Nutzung des Services laufende monatliche Kosten und eine Erhöhung der Komplexität der Softwarearchitektur. PowerSync beispielsweise benötigt zusätzlich zu PostgreSQL auch eine MongoDB Datenbank. Im Gegensatz dazu bleibt bei der Implementierung eines Synchronisierungsmechanismus im Client und der direkten Kommunikation mit dem Backend die Funktionsfähigkeit der App unabhängig von einem solchen externen Dienstleister. Darüber hinaus entstehen keine zusätzlichen Kosten.
Auch wenn die Nutzung eines solchen Synchronisierungsdienstes Vorteile wie die Vereinfachung der Frontendentwicklung und fortschrittliches Konfliktmanagement mit sich bringen würde, überwiegen für die Umsetzung dieser App die Nachteile. Mit Brick wurden während der Entwicklung schnelle Fortschritte erzielt, und im Rahmen der Implementierung der App wurden Beiträge zum Open-Source-Projekt geleistet, von denen die Entwickler-Community in Zukunft profitieren kann.

6.1.4 Google Crashlytics

Um technische Probleme oder Abstürze bei der Benutzung der App zu erfassen, wird Firebase Crashlytics verwendet. Dabei handelt es sich um einen von Google entwickelten kompakten Echtzeit-Absturzreporter, der Stabilitätsprobleme von Apps erfasst, speichert und für Entwickler in einer Weboberfläche nach Auslösern gruppiert. Es hebt die Umstände hervor, die zu den Problemen geführt haben, und erleichtert so die Fehlerbehebung. [Goob].
Die Einbindung eines Services, der lokal auftretende Fehler und Probleme erfasst und dem Entwickler zugänglich macht, wurde getroffen, um die Qualität der App zu erhöhen und sicherzustellen, dass Fehler erkannt und schnellstmöglich behoben

werden können – sowohl während der Entwicklung als auch danach.
Es wurde konkret Firebase Crashlytics gewählt, weil es eine ausgereifte Anbindung zu Flutter bietet und die Probleme in der webbasierten Benutzungsschnittstelle klar anordnet und gruppiert, was die Arbeit mit den Fehlerberichten vereinfacht.

6.1.5 PostgreSQL Datenbank und Supabase

Als Cloud-Datenbank zur Datenspeicherung wurde PostgreSQL gewählt. PostgreSQL ist ein freies, objektrelationales Datenbankmanagementsystem, das SQL erweitert und mit zusätzlichen Funktionen ergänzt. PostgreSQL hat sich einen guten Ruf für seine bewährte Architektur, Zuverlässigkeit, Datenintegrität, seinen robusten Funktionsumfang, seine Erweiterbarkeit und das Engagement der Open-Source-Community hinter der Software erworben. [Pos]
Die Entscheidung für ein relationales SQL Datenbanksystem ist damit begründet, dass das für die zentral gehostete Cloud-Datenbank verwendete System mit der lokal verwendeten Datenbanktechnologie kompatibel sein sollte. Dies liegt daran, dass jede App-Instanz eine eigene lokale Datenbank enthält, die gelegentlich mit der autoritären zentralen Datenbank synchronisiert wird. Da für die lokale Datenbank innerhalb der App-Instanzen SQLite, also ein relationales Datenbanksystem, verwendet wird, ist die Verwendung von PostgreSQL für die zentrale Cloud-Datenbank eine geeignete Wahl. Durch die einheitliche Verwendung relationaler Datenbanksysteme wird die Komplexität der Synchronisation und Migration des Datenbankschemas erheblich reduziert.
Darüber hinaus hat auch die Anforderung des Praxispartners, die Daten bei Bedarf leicht in andere Systeme migrieren zu können, zu dieser Entscheidung beigetragen. Die Verwendung des SQL Standards erleichtert die Migration und Integration der Daten in andere Systeme erheblich.
Die PostgreSQL-Datenbank muss für den Zugriff und die Verwaltung der Daten gehostet werden. Dabei fiel die Wahl auf Supabase, ein Backend-as-a-Service, das auf PostgreSQL basiert. Die Entscheidung für Supabase wurde aus mehreren Gründen getroffen. Erstens stellt Supabase ein auf PostgreSQL basierendes Authentifizierungssystem bereit, das eine sichere und zuverlässige Benutzerverwaltung ermöglicht. Zweitens wird automatisch eine sofort einsatzbereite Representational State Transfer Application Programming Interface (REST-API) auf Basis des Datenbankschemas erstellt, was die Integration und Kommunikation zwischen den verschiedenen Systemkomponenten vereinfacht. Drittens bietet Supabase einen auf Amazon-S3 basierenden Datenspeicher, der für die zukünftige Ergänzung der Fotodokumentation wichtig ist. Diese Funktionen machen Supabase zu einer idealen Wahl für die Anforderungen dieses Projekts.
Als Datenbankstandort für die PostgreSQL Datenbank wurde Frankfurt gewählt, da dieser in Deutschland liegt und von den verfügbaren Serverzentren am nächsten am Nutzungsort in Trier liegt. Supabase nutzt die Amazon Web Services (AWS) zum hosten der Datenbanken.

6.1.6 Supabase Flutter SDK

Zur Kommunikation der App mit dem Supabase Projekt kommt das Supabase Flutter Software Development Kit (SDK) zum Einsatz. Dieses speichert und verwaltet den Authentifizierungsstatus und bietet Methoden zur Authentifizierung und dem Datenbankzugriff an.

6.2 Backend

6.2.1 Datenbankschema

In diesem Abschnitt wird die Datenstruktur der gehosteten PostgreSQL und den SQLite Datenbanken in den Instanzen der App beschrieben, von welcher ein Ausschnitt in Form eines Diagramms in Abbildung 6.2 zu sehen ist.
Die Datenbank besteht aus mehreren Tabellen zur Verwaltung und Speicherung der Daten, die für die Funktionalität und den Betrieb der App erforderlich sind. Diese Tabellen umfassen unter anderem Benutzerinformationen, Kundendaten, Projektdetails und Bestandsaufnahmen von Gebäuden.

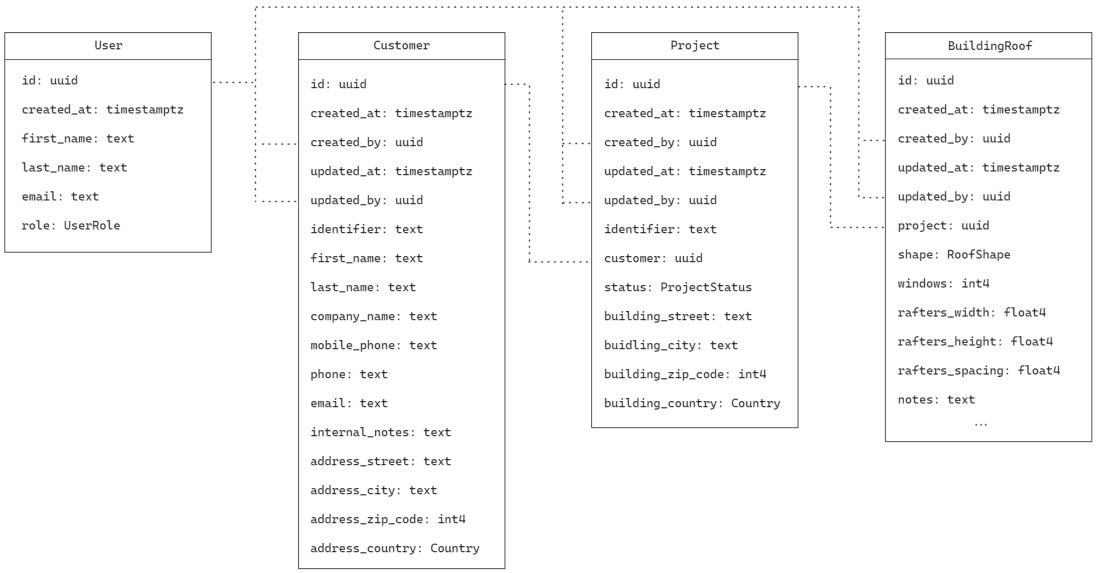


Abbildung 6.2: Ausschnitt des Datenbankschemas inklusive Fremdschlüsselbeziehungen

Grundsätzlich enthalten alle Datenbanktabellen eine Primärschlüssel-Spalte namens „id“ vom Typ UUID (Universally Unique Identifier), welche eine automatisch generierte 128-Bit Zahl zur eindeutigen Identifizierung der Datenreihe enthält. Die Verwendung von UUIDs als Primärschlüssel in der Offline-First-App gewährleistet die globale Eindeutigkeit jedes Objekts, unabhängig vom erstellenden Gerät, und verhindert somit Konflikte bei der Synchronisation.

Im Zentrum der Datenstruktur steht der Entitätstyp der Nutzer. Diese Tabelle hat wie alle anderen Tabellen auch einen englischen Namen, in diesem Fall „users“. Da Supabase eine eigene Tabelle für die Authentifizierung (`auth.users`) verwaltet, enthält die „users“-Tabelle ausschließlich generelle Informationen über den Nutzer, wie den Vor- und Nachnamen, jedoch keine für die Authentifizierung relevanten Daten wie beispielsweise das Passwort. Zur Autorisierung enthält diese Tabelle auch eine „role“-Spalte mit dem Typ eines eigens erstellten Enumerated Type namens `UserRole`. Der Primärschlüssel dieser Tabelle ist die ID des Nutzers, welche gleichzeitig ein Fremdschlüssel zur ID in der `auth.users`-Tabelle ist.
Alle Tabellen zu den Kunden und Projekten enthalten die Spalten „created_at“ und „updated_at“ vom Typ `TimestampTZ` (also einem Zeitstempel inklusive Zeitzone), die speichern, wann der Datensatz erstellt und aktualisiert wurde. Zusätzlich existieren die Metadatenspalten „created_by“ und „updated_by“, die eine Fremdschlüsselbeziehung zur ID in der Nutzertabelle haben.
Die Kundendaten werden in einer Tabelle namens „customers“ gespeichert. Diese enthält unter anderem den Kundennamenn, Kontaktdaten, die Adresse und eine Spalte für interne Notizen.
Eine weitere zentrale Datenbanktabelle ist die „projects“ Tabelle, welche Informationen über Projekte speichert. Diese sind dabei einem Kunden zugeordnet, haben eine Projektnummer, Projektstatus und eine Adresse. Zwischen Kunden und Projekten besteht eine 1:N-Beziehung, jedem Kunden können also beliebig viele Projekte zugeordnet werden und jedes Projekt gehört genau zu einem Kunden.
Für die Gebäudebestandsaufnahme und den Kundeanforderungen zu einem Projekt gibt es mehrere Datenbanktabellen mit dem Präfix „building_“, beispielsweise „building_roof“ für die Daten zum Dach. Diese Tabellen enthalten neben den zuvor erwähnten Standardspalten eine „project_id“-Spalte, die ein Fremdschlüssel zur „projects“-Tabelle ist. Somit können jedem Projekt mehrere Bestandsaufnahmen zu den verschiedenen Bauteilen zugeordnet werden, auch wenn sich die erste Version der App auf eine Bestandsaufnahme pro Projekt beschränkt. Diese Tabellen enthalten vor allem Text-, Bool- und Zahlenfelder. Für Auswahlmöglichkeiten wie zum Beispiel die Dachform wurden eigene Enumerated Types erstellt, die sicherstellen, dass die Daten immer einer festen Auswahl vordefinierter Werte folgen.

Zum Erstellen der Datenbanktabellen und der Enumerated Types wurden SQL Statements geschrieben und im SQL-Editor von Supabase ausgeführt. Listing 6.1 zeigt exemplarisch das Erstellen des Datentyps für die Nutzerrolle.

```
1 CREATE TYPE UserRole AS ENUM(
2     'admin',
3     'internal',
4 );
```

Listing 6.1: SQL Statement zur Erstellung des benutzerdefinierten eigenen Enumerated Types der Nutzerrolle

Wie zuvor beschrieben erstellt Supabase automatisch REST-API-Endpunkte für alle Datenbanktabellen, unter Verwendung von PostgREST. Diese REST-API wird im Rahmen des Projekts zum Abfragen und Speichern der Daten durch die Flutter App verwendet.

6.2.2 Authentifizierung

Wie bereits erwähnt, bietet Supabase einen Authentifizierungsservice, der eine API zur Erstellung und Verwaltung von Nutzern anbietet. Dieser Service verwendet die PostgreSQL-Datenbank des Projekts und speichert Benutzerdaten sowie andere Authentifizierungsinformationen in einem speziellen `auth`-Schema. Diese Daten können unter anderem mithilfe von Fremdschlüsselverweisen mit eigenen Tabellen verbunden werden.
Beim Einloggen eines Nutzers in der App über die Supabase Dart-SDKs wird ein Access-Token, genauer gesagt ein JSON Web Token (JWT), erzeugt. Dieser Token enthält Identitäts- und Authentizitätsinformationen über den Nutzer sowie Informationen über die Laufzeit und wird mit einem kryptografischen Schlüssel signiert, um ihn manipulationssicher zu machen. Bei jeder Anfrage an einen Supabase-Service wird dieser Access-Token mitgesendet [Sup].
Für die Anwendung wurde im Supabase Dashboard eine E-Mail- und Passwort-Authentifizierung aktiviert sowie Vorlagen eingerichtet, die beim Versand von E-Mails zum Wählen oder Ändern des Passworts verwendet werden.
Für die Speicherung der Nutzerdaten wie Name und Rolle wurde zunächst die von Supabase verwaltete `auth.users` Tabelle genutzt. Da die Struktur dieser nicht geändert werden kann, enthält diese eine Spalte vom Typ `JSONB` für Metadaten über den Nutzer, die von der Dart-SDK im Client gesetzt und abgerufen werden kann. Obwohl diese Methode funktioniert hat, weist sie zwei Nachteile auf: Erstens werden die Daten in der `JSONB`-Spalte gespeichert, wodurch auf Datenbankebene kein festes Schema sichergestellt werden kann. Darüber hinaus könnten diese Informationen clientseitig geändert werden, was insbesondere bei der Nutzerrolle nicht erwünscht ist.
Aus diesen Gründen wurde stattdessen eine eigene „users“ Datenbanktabelle erstellt, die mit Fremdschlüsselverweisen mit der `auth.users` Tabelle verknüpft ist. Diese ist so konfiguriert, dass die Daten nur von Nutzern der Rolle Administrator geändert werden können.

6.2.3 Benutzerverwaltung

Benutzerkonten für die Mitarbeiter sollen in der App bei Bedarf ausschließlich von Administratoren erstellt werden können, um sicherzustellen, dass nur ausgewählte Personen Zugriff auf die App haben.
Das Erstellen und Löschen von Nutzerkonten über die API des Supabase Authentifizierungsservice ist aus Sicherheitsgründen nur innerhalb einer Serverumgebung möglich. Um dies umzusetzen, wurden Supabase Edge Functions implementiert. Diese Funktionen, die in der JavaScript-Laufzeitumgebung Deno entwickelt wurden, sind in Rechenzentren nahe am Nutzer verfügbare Funktionen und können

vom Client aufgerufen werden [Sup].
Zwei Funktionen wurden in TypeScript entwickelt: Eine zum Erstellen und eine zum Löschen eines Nutzerkontos. Diese Funktionen werden vom Client zusammen mit dem Access-Token aufgerufen. Innerhalb dieser Funktionen wird überprüft, ob der Access-Token gültig ist und ob der zugehörige Nutzer ein Administrator ist. Falls dies der Fall ist, wird das Nutzerkonto zunächst im Supabase Authentifizierungsservice erstellt/gelöscht und anschließend in der `public.users`-Tabelle eingetragen/gelöscht.

6.2.4 Autorisierung

Wie zuvor beschrieben, erlaubt PostgreSQL das Erstellen von Row Security Policies, auch Row Level Security (RLS) genannt, die es ermöglichen, auf Nutzerbasis zu steuern, welche Reihen gelesen, geschrieben, aktualisiert oder gelöscht werden dürfen. Durch das Aktivieren von RLS für eine Tabelle müssen alle Aktionen explizit durch eine entsprechende Regel erlaubt sein. Bei jedem Zugriff auf eine Reihe wird ein Ausdruck evaluiert, der entweder wahr oder falsch zurückliefert und damit signalisiert, ob der Nutzer die Reihe sehen oder bearbeiten darf. [Pos]
Um eine auf Nutzerrollen basierende Autorisierung umzusetzen, wurden für jede Datenbanktabelle eine oder mehrere Regeln erstellt. Beispielsweise enthält die Projekttabelle eine Regel, die nur authentifizierten Nutzern mit der Rolle Administrator oder interner Mitarbeiter erlaubt, neue Projekte zu erstellen. Das SQL-Statement für die Erstellung dieser Regel ist in Listing 6.2 dargestellt.

```
1  CREATE POLICY "Authenticated␣users␣with␣admin␣or␣internal␣role␣can␣insert"
2  ON "projects"
3  FOR INSERT
4  TO AUTHENTICATED
5  WITH CHECK (
6      auth.uid() IN (
7         SELECT id
8         FROM users
9         WHERE role = ANY (ARRAY['admin'::"UserRole", 'internal'::"UserRole"])
10     )
11 );
```

Listing 6.2: SQL Statement zum Erstellen einer RLS Regel für die Projekttabelle

6.2.5 Produktions- und Entwicklungsumgebung

Für die Entwicklung und den Betrieb der Anwendung wurden zwei separate Backendumgebungen inklusive eigener Datenbank eingerichtet: eine Umgebung, welche beim Entwickeln und Testen benutzt wird (genannt `DEV`) und eine, die von den Instanzen der App die beim Praxispartner verwendet wird (genannt `PROD`).
Beim Kompilieren der App kann über eine Konfiguration mitgegeben werden, mit welcher Backendumgebung diese kommunizieren soll. Diese Trennung der Umgebungen ermöglicht es, neue Funktionen und Änderungen in einer kontrollierten Umgebung zu testen, bevor sie in die produktive Umgebung übernommen werden.

6.3 Frontend

Nachdem im vorherigen Abschnitt das Backend erläutert wurde, wird in diesem Abschnitt die Entwicklung des Frontends, also des Clients, der Flutter App, beschrieben.

6.3.1 Entwicklungsworkflow

Zur Entwicklung der Flutter App wurde zunächst ein Git-Repository erstellt und bei GitHub gehostet. Dies ermöglicht eine Versionsverwaltung und bietet weitere Tools wie Branches und Pull Requests für eine effiziente Entwicklung.
Zur Automatisierung wurde eine Continuous Integration und Delivery (CI/CD) Pipeline eingerichtet. Dafür wurden GitHub Actions Workflows verwendet. Diese Workflows sind automatisierte Prozesse, die im Repository in einer YAML-Datei gespeichert sind und ein oder mehrere Jobs auf Basis von Triggern oder Events innerhalb einer virtuellen Maschine ausführen [Git].
Für dieses Projekt wurden drei Workflows erstellt. Der erste Workflow wird bei jedem Commit in den Main Branch ausgeführt und besteht aus drei Jobs: zunächst wird der Flutter-Code analysiert, anschließend die Web-Version gebaut und schließlich im Web gehostet. Dieser Workflow ermöglicht es, Änderungen automatisiert zu prüfen, zu bauen und auszuliefern.
Der zweite Workflow ist ähnlich aufgebaut, wird jedoch bei jedem neuen Pull Request ausgeführt. Dabei wird die gebaute Web-App nicht auf die Produktions-URL deployed, sondern es wird eine Preview-URL erstellt, über die die geänderte Software manuell getestet werden kann.
Der dritte Workflow dient dem Bau der iOS-Version der App, startet einen iOS-Simulator und führt die entwickelten Integrationstests der App aus. Dieser Workflow wird ebenfalls bei jedem Pull Request ausgeführt und stellt sicher, dass Änderungen am Code bestehende Funktionen nicht beeinträchtigen.

6.3.2 Bereitstellung der App

Zur Automatisierung des Bauens und Auslieferns der App werden verschiedene Tools und Services eingesetzt. Wie zuvor beschrieben wurde eine GitHub Action eingerichtet, die automatisch die Web-App baut und an den Webhosting-Anbieter sendet. Dieser Prozess stellt sicher, dass jede Änderung am Code sofort überprüft und die aktuelle Version der Web-App bereitgestellt wird.
Für die iOS-App wird der Service Codemagic verwendet, welcher den Bau der iOS-App übernimmt und sendet diese entweder an TestFlight für die Entwicklungsphase oder an App Store Connect für die Veröffentlichung im Apple App Store. Während der Entwicklungsphase ermöglicht TestFlight ein einfaches Testen und Verteilen der App an Tester. Nach Abschluss der Tests und der Freigabe wird die App dann im Apple App Store veröffentlicht. Eine Schwierigkeit hierbei war es, die Verknüpfung zur Apple App Store Connect REST-API vorzunehmen, wodurch die nötigen Zertifikate und Profile zur Codesignierung automatisch übertragen werden.

6.3.3 Softwarearchitektur

Die Implementierung der Flutter-App folgt einem strukturierten und modularen Ansatz. Die Architektur der App basiert auf der in Abbildung 6.3 skizzierten Trennung von Präsentation, Zustand und Datenebene, um eine klare Struktur und Wartbarkeit des Codes zu gewährleisten.

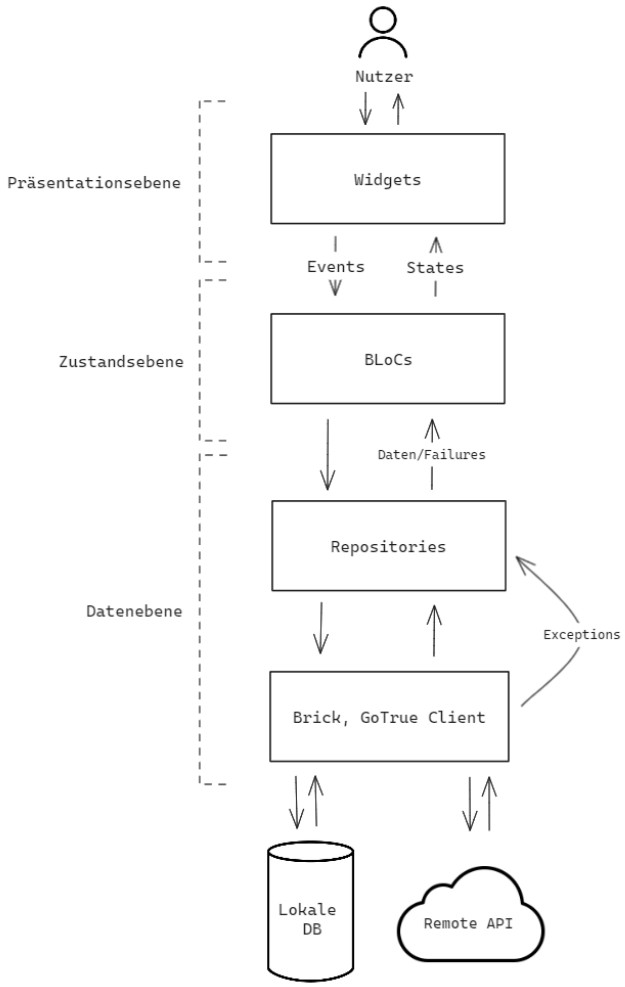


Abbildung 6.3: Zusammenspiel der drei Frontend-Ebenen

Präsentationsebene

Die Präsentationsebene bildet die Benutzeroberfläche der App ab und ist für die Interaktion mit dem Benutzer zuständig. Diese Ebene nutzt die vom Flutter-Framework bereitgestellten Klassen zur Erstellung von UI-Komponenten in Form von sogenannten `Widgets`. Da der Zustand der App in der gesonderten Zustandsebene gespeichert und verwaltet wird, werden diese UI-Komponenten ausschließlich von der `StatelessWidget` Klasse abgeleitet.

Zustandsebene

Die Zustandsebene verwaltet den Zustand der Anwendung und die Geschäftslogik, die diesen Zustand verändert. Für die Zustandshandhabung wird das Flutter BLoC (Business Logic Component) Pattern verwendet. Dieses Pattern ermöglicht eine klare Trennung zwischen der Präsentationslogik und der Zustandslogik, wodurch die App besser testbar und wartbarer wird. Die Bloc-Komponenten übernehmen die Aufgabe, auf Benutzerinteraktionen zu reagieren und den Zustand entsprechend zu aktualisieren. Die Objekte zur Speicherung des Zustandes sind dabei unveränderlich (engl. „immutable“), der Zustand kann nur durch das Erzeugen einer neuen Objektinstanz mithilfe einer `copyWith` Methode geändert werden. Dies hat den Vorteil, dass Änderungen am Zustand leicht festgestellt werden können und keine Seiteneffekte am Zustandsobjekt möglich sind.

Datenebene

Die Datenebene, auch als Store-Ebene bezeichnet, ist für die Persistenz und Verwaltung der Daten zuständig. Innerhalb der Datenebene gibt es Repositories, die als Schnittstelle zwischen den BLoCs und den Datenquellen dienen. Es gibt ein generisches Datenrepository, das für alle Models genutzt werden kann und das spezifische Modell als generischen Parameter übergeben wird. Ebenso existiert ein Authentifizierungs-Repository. Die Repositories bieten Methoden an, die von den BLoCs genutzt werden, um Daten zu speichern, zu laden oder Authentifizierungsprozesse durchzuführen. Diese Methoden greifen dann auf Brick oder den GoTrue Client zur Authentifizierung zu. Dabei können Ausnahmen (Exceptions) auftreten, beispielsweise, wenn der Zugriff auf die Remote-REST-API fehlschlägt. Diese werden vollständig auf der Repository-Ebene abgefangen (Catch) und in Failure-Enums umgewandelt. Jede Methode der Repositories hat den Rückgabetyp `Either`, ein Typ aus der funktionalen Programmierung, der zwei Seiten hat: eine Seite, die im Erfolgsfall die Daten hält, und eine, die den aufgetretenen Fehler enthält. Ein Beispiel ist der Rückgabetyp einer Methode des Nutzerdatenrepositories: `Either<FetchFailure, List<User>>`. In der Zustandsebene kann dann basierend auf Erfolgs- oder Fehlschlagsfall entsprechend der App-Zustand geändert werden. Dieses Abfangen und Behandeln der Ausnahmen auf Datenebene stellt sicher, dass es zu keinen unbehandelten Ausnahmen kommt, was Anzeigefehler, Funktionsfehler oder sogar Abstürze der App verhindert, die zu einem schlechten Nutzererlebnis und einer verminderten Gebrauchstauglichkeit führen würde.
Innerhalb der Datenrepositories kommt das Brick-Package zum Einsatz, das eine effiziente und strukturierte Art und Weise bietet, Datenmodelle zu definieren und Datenoperationen zu verwalten. Es ist dafür verantwortlich, Daten zunächst aus der lokalen Datenbank zu beziehen sowie diese darin abzuspeichern und bei Bedarf mit der Remote-Datenbank zu kommunizieren.
Für das Authentifizierungsrepository wird auf unterster Ebene der vom Supabase Dart-SDK bereitgestellte GoTrue Client verwendet. Dieser bietet Methoden zur Authentifizierung und führt automatisch im Hintergrund die Aktualisierung des Access-Tokens durch.

Fazit

Durch diese klare Trennung der Ebenen wird eine hohe Modularität und Wiederverwendbarkeit des Codes erreicht. Jede Ebene hat ihre klar definierten Verantwortlichkeiten, was die Entwicklung, Wartung und Erweiterung der App vereinfacht. Teile der Geschäftslogik und des Zustandsmanagements sind funktional geschrieben, was eine präzisere und weniger fehleranfällige Handhabung von Zustandsänderungen ermöglicht. Auf Präsentationsebene führt die Trennung zwischen Präsentation und Zustand zu weniger Rendervorgängen der GUI zur Laufzeit, da nur die Widgets neu gebaut werden müssen, bei denen sich die zugrunde liegenden Teile des Zustandes geändert haben.

6.3.4 Formulare

Anforderungen

Da eine Kernfunktionalität der App die Formulare sind und an diese besondere Anforderungen gestellt werden, ist die Umsetzung dieser im Frontend ein komplexer Prozess. Die folgenden Anforderungen bestehen an die Formulare:

- **Verschiedene Werttypen**: Unterstützung für verschiedene Datentypen wie Wahrheitswert, Ganzzahl, Kommazahl, Aufzählung und Text.
- **Gruppierung von Formularfeldern**: Möglichkeit, Formularfelder in Gruppen zu organisieren.
- **Duplizierung von Formulargruppen**: Fähigkeit, Formulargruppen zu duplizieren, um eine Liste gleichartiger Formulargruppen zu erstellen.
- **Formularfeldabhängigkeiten**: Bestimmte Felder oder Gruppen sollen nur angezeigt oder ausgefüllt werden, wenn ein anderes Feld einen bestimmten Inhalt hat.
- **Validierung**: Eingaben müssen überprüft werden, z. B. auf gültige E-Mail-Adressen, Postleitzahlen oder Zahlenwerte.

Flutter bietet zur Erstellung von Formularen Widgets zum Rendern der Formularfelder (Textfeld, Dropdown, Checkbox etc.) an und auch ein Widget zum Gruppieren dieser Felder in ein Formular, um die Speicherung und Validierung der Felder durchzuführen. Diese primitiven Widgets sind jedoch nicht für die komplexen oben genannten Anforderungen ausgelegt und würden zu unübersichtlichem Code und erhöhtem Fehlerpotential führen, besonders vor dem Hintergrund, dass die App aus vielen großen Formularen besteht und diese eine essenzielle Komponente der App darstellen. Stattdessen wurde ein eigenes Flutter-Package namens Formy entwickelt, das alle oben genannten Anforderungen erfüllt. Dies ermöglicht es, mit wenig Aufwand neue Formulare zu erstellen oder vorhandene zu ändern. Das Package abstrahiert die benötigten Funktionen und bietet eine zielgerichtete API, die die Entwicklung und Wartung von vielen Formularen mit verschiedenen Schemas erheblich erleichtert.

Modellbasierter Ansatz

Bei der Eigenentwicklung des Formular-Packages Formy wurde ein modell- bzw. klassenbasierter Ansatz zur Handhabung von Formularen verwendet, da die meisten der Formulare in der App auf einem Modell basieren, welches aus der Datenbank geladen, bei Bedarf geändert und dann wieder gespeichert werden soll.
Für die Verwaltung des Formularzustands wird das BLoC Pattern verwendet. Formy stellt hierfür die folgenden Klassen zur Verfügung:
Die Klasse `FormyField` speichert den Zustand eines Formularfelds. Zu den gespeicherten Informationen gehören der Wert des Felds, das UI-Label, der Bearbeitungszustand (bearbeitet/unbearbeitet, pure und dirty genannt), die Validatoren, die Validierungsnachricht und ob das Feld aktiviert oder deaktiviert ist. Die Klasse ist generisch und bestimmt so den Datentyp des Wert-Attributs.
Zusätzlich gibt es die abstrakte Klasse `FormyValidator`, die eine Methode zur Validierung von Eingabewerten und Rückgabe einer Nachricht enthält. Diese wird von spezifischen Validatoren wie `RequiredValidator`, `MinValueValidator` oder `EmailValidator` implementiert.
Eine weitere Komponente ist die Klasse `FormyGroup`, die eine Sammlung von Formularfeldern verwaltet. Diese Klasse enthält eine Map, die Formularfeld-Schlüsseln den entsprechenden `FormyField`-Instanzen zuordnet. Um die im Formular erfassten Daten speichern zu können, bietet es außerdem eine Methode zum Umwandeln der Formulardaten in eine Instanz der generischen Modell-Klasse.
Die Klasse `Formy` ist die zentrale Komponente des Packages und speichert eine `FormyGroup`-Instanz sowie den Zustand der Formularübermittlung. Sie enthält außerdem ein Attribut namens `onSubmit`, welches ausgeführt wird, wenn das Formular abgesendet wurde. Diese Struktur ermöglicht es, den gesamten Prozess der Formularerstellung, -validierung und -übermittlung effizient und strukturiert zu verwalten.
Da alle diese Klassen zusammen den Zustand des Formulars repräsentieren und im Rahmen des BLoC Pattern benutzt werden, sind sie unveränderbar und verfügen über eine `copyWith` Methode zum Erzeugen neuer Objekte auf Basis von vorhandenen Objekten. Im Rahmen dieser App wird für die Implementierung des BLoC Patterns das `flutter_bloc` Package und dessen `Cubit` Klasse verwendet. Die Klasse `FormyCubit` verwaltet Zustandsänderungen und über diese können die Präsentations-Widgets mit Formy kommunizieren, beispielsweise zur Änderung des Wertes eines Formularfelds.
Neben diesen reinen Dart-Klassen existieren auch Widgets, die etwa das Layout eines Formulars festlegen und die Möglichkeit bieten eigene Widgets für die Präsentation zu nutzen.

Verwendung

Zur Erstellung eines Formulars wird zunächst eine `FormyGroup` Klasse erstellt. In Listing 6.3 ist dies exemplarisch ausschnittsweise für die Bestandsaufnahmedaten zum Keller gezeigt. Dabei wird über den generischen Parameter angegeben,

für welches Modell bzw. welche Dart-Klasse die Formulargruppe erstellt werden soll. Das `toModel` Attribut spezifiziert eine Funktion, die als Parameter unter anderem die Werte aller Formularfelder erhält und diese in eine entsprechende Instanz der gewählten Klasse umwandelt. Zudem wird das Formularschema durch das `formFields` Attribut definiert. Darin kann auch angegeben werden, welche Bedingungen erfüllt sein müssen, damit ein Feld aktiviert oder deaktiviert werden soll.

```
final formyGroup = FormyGroup<BuildingBasement>(
    name: 'buildingBasement',
    databaseId: initialData.id.value,
    toModel: (formFields, _, databaseId) => BuildingBasement(
        id:databaseId,
        projectId: projectId,
        hasBasement: formFields['hasBasement']!.value as bool?,
        conditioning: formFields['conditioning']!.value
                                                    as BasementConditioning?
        ...
    ),
    formFields: {
        'hasBasement': FormyField<bool?>.pure(
            label: 'Keller vorhanden',
            value: initialData.hasBasement,
        ),
        'conditioning': FormyField<BasementConditioning?>.pure(
            label: 'Konditionierung',
            value: initialData.conditioning,
            disabled: (formFields) => formFields['hasBasement']!.value
                                                                != true,
        ),
        ...
    }
);
```

Listing 6.3: Instantiierung eines FormyGroup Objekts für das Kellerdatenformular

Die Struktur der Formulargruppe ist damit in ein `FormyGroup` Objekt eingebettet. Dieses kann nun verwendet werden, um ein `Formy` Objekt zu erzeugen. Dabei wird zusätzlich eine Funktion übergeben, die automatisch aufgerufen wird, wenn das Formular abgesendet wird. Im Rahmen der App wird in diesem Fall die `upsert` Methode auf dem entsprechenden Repository aufgerufen, um die Daten zu speichern. Das `Formy` Objekt handhabt dann die Verwaltung des Zustands der Formularfelder und der Speicherung des Formulars.

```
final formy = Formy<BuildingBasement>(
    formGroup: formyGroup,
    onSubmit: (data) => GenericRepository<BuildingBasement>.upsert(data)),
),
```

Listing 6.4: Instantiierung eines Formy Objekts für das Kellerdatenformular

Innerhalb eines Flutter-Widgets in der Präsentationsebene kann das erzeugte `Formy` Objekt nun genutzt werden, um zum Formularzustand die entsprechenden Widgets zur Darstellung des Formulars anzuzeigen. Dazu gehören beispielsweise von Flutter bereitgestellte Texteingabefelder, Auswahlfelder und ein Button zum Absenden des Formulars. Alle diese Widgets reagieren auf den Zustand des

Formulars, welcher mit einem `BlocProvider` und `FormyCubit` in Flutters Widget-Baumstruktur den Widgets zugänglich gemacht wird.
Zur weiteren Abstraktion auch in der Präsentationsebene wurden zwei Widgets namens `FormyWidget` und `FormGroupLayout` erstellt, die die zuvor genannten Aufgaben übernehmen. Diese Widgets erzeugen automatisch basierend auf dem Formularschema die entsprechenden UI-Komponenten. Für jedes Formular auf der Präsentationsebene muss daher nur angegeben werden, in welchem Layout die Felder angezeigt werden sollen, was über eine `fieldKeys` Listenmatrix erfolgt.

```
FormyWidget<BuildingBasement>(
    initialData: formData,
    formyCubit: FormyCubit<BuildingBasement>(formy: formy),
    child: Column(
        children: [
            const FormGroupLayout<BuildingBasement>(
                title: 'Allgemein',
                fieldKeys: [
                    ['hasBasement', 'conditioning'],
                ],
            ),
        ],
    ),
)
```

Listing 6.5: Erstellung des Formulars auf Präsentationsebene

Durch die Implementierung dieser eigenen Formular-Bibliothek und der damit verbundenen Abstraktion ist es möglich, viele Formulare mit den oben genannten Anforderungen zu erstellen. Gleichzeitig wird sichergestellt, dass dabei keine großen Mengen an sich wiederholendem Code entstehen.

Handschrifterkennung

Die Handschrifterkennung, also die Umwandlung von handschriftlich geschriebenem in Text innerhalb der Formularfelder, wird in der iOS-App durch die native iPadOS-Funktion „Kritzeln“ automatisch bei der Verwendung eines Apple Pencils unterstützt. Für diese Funktion war daher kein zusätzlicher Implementierungsaufwand erforderlich.

6.3.5 Offlinefähigkeit und Synchronisierung

Eine zentrale Anforderung an die App ist die Fähigkeit, Kunden- und Projektdaten auch ohne eine Internetverbindung erstellen und bearbeiten zu können. Das Supabase Dart SDK bietet jedoch nur eine Funktion für die Speicherung des Authentifizierungsstatus und des aktuellen Access Keys, aber keine Möglichkeit, die Datenbankinhalte lokal zu persistieren und damit auch ohne Internetzugriff zugänglich und bearbeitbar zu machen.
Um dieser Anforderung gerecht zu werden, wurde das oben beschriebene Brick-Package gewählt, das einen Offline-First-Ansatz bietet. Dafür wird eine lokale SQLite-Datenbank mit Tabellen für alle Modelle, die das gleiche Schema wie die Remote-Datenbank haben, verwendet.

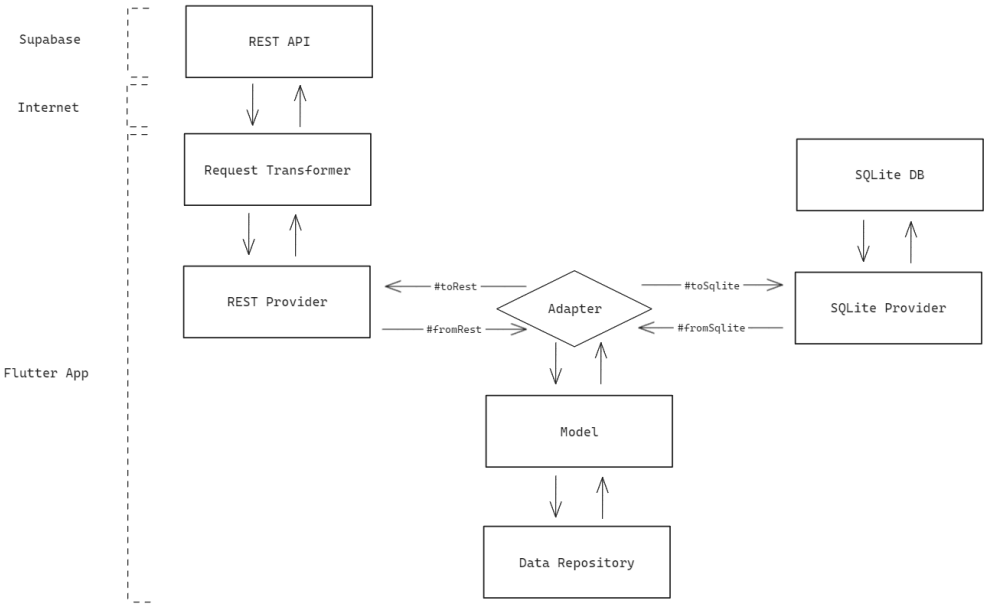


Abbildung 6.4: Brick Komponenten und Architektur

Der Aufbau von Brick ist in Abbildung 6.4 zu sehen. Das `DataRepository` verwaltet die Datenoperationen zum Speichern und Laden der Daten und ist generisch, sodass für jedes Modell ein eigenes Repository erstellt werden kann. Modelle sind Dart-Klassen, die mit Brick-spezifischen Dart-Annotationen versehen werden, auf deren Basis automatisch Adaptermethoden generiert werden. Diese Methoden wandeln Instanzen der Klasse in REST/SQLite-kompatible JSON-Objekte um, die dann in der lokalen oder Cloud-Datenbank gespeichert werden können. Für jedes Modell existiert eine REST-Request-Transformer-Klasse, die über die Methoden `get`, `delete` und `upsert` verfügt und auf Basis des Modells und der Query den Pfad zum REST-API-Endpunkt zurückliefert. [Cou]
Da Brick die Verwendung des Supabase Dart-SDK als Remote Provider nicht unterstützt, wurde stattdessen die von Supabase automatisch generierte REST-API verwendet. Um dies zu ermöglichen, muss Brick so konfiguriert werden, dass beim Versenden der Anfragen an die REST-API die Authentifizierungs-Header mitgesendet werden, damit die Anfragen nicht abgelehnt werden. Dies wurde während der Entwicklung zunächst so umgesetzt, dass die Authentifizierungs-Header beim Start der App an die Repository-Klasse von Brick übergeben wurden und so bei jedem Request mitgesendet wurden. Obwohl dies beim Testen während der Entwicklung funktionierte, stellte sich später heraus, dass dies nicht sinnvoll ist, da eine Anfrage möglicherweise erst Stunden nach der Änderung in der lokalen Datenbank versendet werden kann, wenn eine Internetverbindung hergestellt wird. Die REST-API würde dann den Request ablehnen, da der JWT, der im Header mitgesendet wird, abgelaufen und nicht mehr gültig ist.
Um dieses Problem zu beheben, wurde stattdessen eine Klasse namens `JWTClient`

erstellt, die einen HTTP `BaseClient` erweitert und die `send` Methode, wie in Listing 6.6 zu sehen, überschreibt. Dadurch wird dem Request-Header der aktuell gültige Access Key und der API-Key hinzugefügt, bevor die `send` Methode des inneren HTTP Clients ausgeführt wird, der den Request dann versendet. Wenn Brick so konfiguriert ist, dass dieser HTTP-Client für die REST-Requests verwendet wird, ist sichergestellt, dass die Authentifizierungs-Header zu jedem Zeitpunkt gültig sind und es zu keinen Problemen bei der Kommunikation mit der REST-API kommt.

```
1  @override
2  Future<http.StreamedResponse> send(http.BaseRequest request) async {
3      final accessToken =
4                Supabase.instance.client.auth.currentSession?.accessToken;
5
6      request.headers.addAll({
7          if (accessToken != null) 'Authorization': 'Bearer $accessToken',
8          'apikey': SUPABASE_ANON_KEY
9      });
10
11     return _innerClient.send(request);
12 }
```

Listing 6.6: send Methode des JWTClients für die Kommunikation mit der REST-API

Beim Abfragen der Daten vom `DataRepository` bietet Brick das Angeben eines `Policy`-Enums der angibt, ob die Daten nur von der lokalen Datenbank oder (auch) von der Remote Datenbank (falls möglich) zurückgeliefert werden sollen. Innerhalb dieses Projektes werden die Daten bei allen Formularen zunächst von der lokalen Datenbank geladen, dann in der UI angezeigt und dann im Hintergrund versucht, die Daten von der Cloud Datenbank abzufragen. Währenddessen erscheint ein Hinweis, dass die Daten gerade geladen werden; falls dies nicht erfolgreich ist, sieht der Nutzer einen Hinweis, dass er offline ist und die aktuellen Daten möglicherweise veraltet sind.

Synchronisierung

Um sicherzustellen, dass lokale Datenänderungen, beispielsweise bei der Bestandsaufnahme ohne aktive Internetverbindung, zu einem späteren Zeitpunkt in der Cloud-Datenbank gespeichert werden und dadurch für andere Nutzer oder auf anderen Geräten verfügbar sind, synchronisiert Brick diese Daten.
Bei jeder Änderung (beispielsweise das Hinzufügen oder Ändern eines Projekts) wird diese zunächst in der lokalen SQLite-Datenbank ausgeführt und gespeichert. Zusätzlich wird diese in einen HTTP-Request an die REST-API umgewandelt und in einer gesonderten Datenbanktabelle gespeichert. Dabei werden unter anderem die folgenden Werte gespeichert: das Erstellungsdatum, die API-URL und der Body (mit den eigentlichen gespeicherten Daten) sowie die Anzahl der erfolglosen Versuche den Request zu versenden.
Im Hintergrund wird mithilfe eines Timers regelmäßig, standardmäßig alle 5 Sekunden, versucht den ältesten Request an die REST-API zu senden. Genau gesagt

ist dies in Brick als Queue mit serieller Abarbeitung umgesetzt. Um zu bestimmen, ob ein Request erfolgreich war und aus der Queue entfernt werden kann oder erneut versucht werden muss, wird der HTTP-Statuscode des Server-Response verwendet.
Da Brick ein reines Dart- und kein Flutter-Package ist, muss jegliche Benachrichtigung des Nutzers über den Synchronisierungsstatus eigenständig implementiert werden. Um dem Nutzer zu signalisieren, ob es unsynchronisierte Änderungen gibt, wurde eine Funktion geschrieben, die alle Requests aus der lokalen Offline-Queue in einen eigens erstellten Dart-Typ umwandelt. Dadurch konnte eine Seite erstellt werden, die eine Liste aller ausstehenden Änderungen (Requests) enthält, sodass der Nutzer sehen kann, welche Änderungen ausstehen.
Zusätzlich wurde ein Mechanismus implementiert, um den Synchronisierungsstatus zu erkennen, da dies mit Brick nicht möglich ist. Dieser Mechanismus bestimmt, ob die Synchronisierung gerade durchgeführt wird, das Gerät offline ist oder eine Anfrage vom Server abgelehnt wurde. Auf diese Weise wird der Nutzer über den aktuellen Status der Synchronisierung informiert.

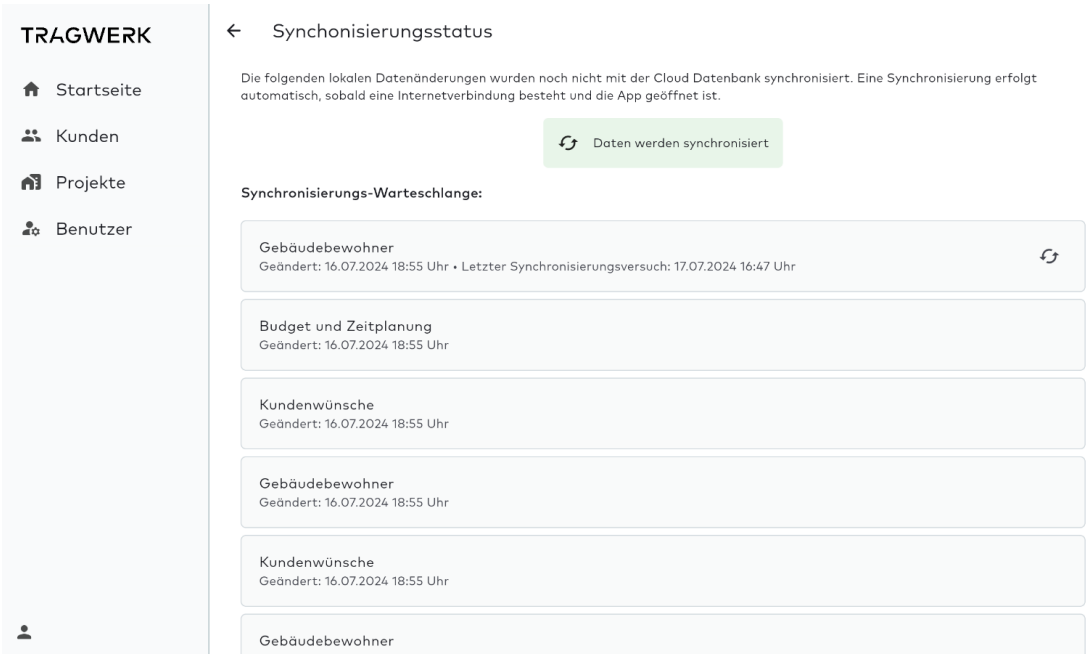


Abbildung 6.5: Screenshot der Synchronisierungsstatus Seite während der Synchronisierung

Synchronisierungskonflikte

Bei der Synchronisierung von offline erstellten oder bearbeiteten Daten kann es durch den Charakter des verteilten Systems zu Konflikten kommen, wenn ein anderer Nutzer ebenfalls Änderungen am gleichen Datensatz, beispielsweise einem Projekt, vorgenommen hat.

Obwohl dies bei der Benutzung der entwickelten App unwahrscheinlich ist, da die Nutzergruppe vergleichsweise klein ist und Daten nur im Rahmen der Bestandsaufnahme einmalig offline erstellt werden, wurde dieser Aspekt bei der Entwicklung berücksichtigt. Allerdings konnte er aufgrund des begrenzten Zeitrahmens nicht vollumfänglich behandelt werden. Dazu könnten, wie im Artikel von Kleppmann aus Kapitel 2.1.3 vorgeschlagen, Conflict-free Replicated Data Types (CRDTs) verwendet werden. Dies überstieg jedoch den Rahmen dieser Arbeit.
Ein Beispiel für eine Konfliktsituation mit potenziellem Datenverlust wäre, wenn Nutzer A die App offline benutzt und einen bestehenden Datensatz ändert, während Nutzer B online denselben Datensatz bearbeitet.
Um zu verhindern, dass in solchen Fällen lokal bearbeitete Datensätze, die auf einem veralteten Stand beruhen, den aktuellen Stand in der Cloud-Datenbank überschreiben, wird zu jedem Datensatz gespeichert, wann dieser zuletzt lokal bearbeitet wurde. Dies ermöglicht es, beim Senden der Daten an die Datenbank zu bestimmen, ob zwischenzeitlich ein neuerer Stand der Daten existiert. Um auf Datenbankebene Datensätze abzulehnen, deren Aktualisierungsdatum älter ist als das des aktuell in der Cloud-Datenbank gespeicherten Objekts, wurde eine PostgreSQL-Funktion implementiert, die eine Exception wirft, wenn das `updated_at` Feld des zu speichernden Objekts älter ist als das des aktuell gespeicherten Objekts:

```
1  CREATE FUNCTION check_updated_at()
2  RETURNS TRIGGER AS $$
3  BEGIN
4      IF NEW.updated_at < OLD.updated_at THEN
5          RAISE EXCEPTION
6              'New␣updated_at␣must␣be␣after␣the␣old␣updated_at.';
7      END IF;
8      RETURN NEW;
9  END;
10 $$ LANGUAGE plpgsql;
```

Listing 6.7: PostgreSQL-Funktion, die eine Exception wirft, wenn das updated_at Feld des zu speichernden Objekts älter ist als das des aktuell gespeicherten Objekts

Diese Funktion wird dann mithilfe eines `BEFORE UPDATE-Trigger`s vor jeder Aktualisierung bei allen Datenbanktabellen ausgeführt:

```
1  CREATE TRIGGER check_updated_at_trigger
2  BEFORE UPDATE ON table_name
3  FOR EACH ROW
4  EXECUTE FUNCTION check_updated_at();
```

Listing 6.8: Erstellung eines BEFORE UPDATE-Triggers unter Verwendung der zuvor erstellen Funktion

Somit ist sichergestellt, dass alle Update-Anfragen an die Datenbank abgewiesen werden, die neuere Daten überschreiben würden, wodurch ein Datenverlust in vergleichbaren Situationen vermieden wird.

Softwarequalitätssicherung

Zur Sicherung der Softwarequalität und der Erfüllung der festgelegten Nutzungsanforderungen wurden umfassende Softwaretests entwickelt. Diese Tests stellen insbesondere die Zuverlässigkeit des Systems sicher, indem überprüft wird, ob die spezifizierten Funktionen unter den festgelegten Bedingungen ordnungsgemäß arbeiten. Hierbei wurde die Verwendung von End-to-End-Tests (E2E-Tests) gewählt, um das System von Anfang bis Ende zu testen und sicherzustellen, dass die integrierten Komponenten ordnungsgemäß zusammenarbeiten.
Das Flutter SDK enthält ein Package namens `integration_test`, das selbstgesteuerte Tests von Flutter-Code auf Geräten und Emulatoren ermöglicht. Da eine Hauptanforderung die Offlinefähigkeit des Systems ist, muss auch diese Funktionalität durch die Tests verifiziert werden. Um dies zu erreichen, muss der Test auf die nativen Plattformfunktionen zum Aktivieren und Deaktivieren des WLANs und der mobilen Datenverbindung des Geräts zugreifen, auf dem die Flutter-App läuft. Da dies mit dem `integration_test`-Package nicht möglich ist, wurde im Rahmen der Softwaretests das Flutter Package Patrol verwendet.
Mit Patrol wurden Tests erstellt, die alle Hauptanforderungen der App überprüfen, indem die App auf einem virtuellen Testgerät gestartet wird und das Nutzungsverhalten eines Nutzers simuliert. Dies geschieht durch automatisierte Navigation durch die App und Eingaben, um abschließend zu prüfen, ob sich das System im erwarteten Endzustand befindet. Dabei wird beispielsweise ein Formular zur technischen Gebäudebestandsaufnahme offline ausgefüllt. Anschließend wird geprüft, ob die Daten erfolgreich gespeichert wurden und nach erneutem Öffnen des Formulars exakt die zuvor eingegebenen Daten erscheinen. Diese Softwaretests kommunizieren ausschließlich mit der Entwicklungs-Backendumgebung, um die Funktionalität und Zuverlässigkeit der App in einer kontrollierten Umgebung zu testen.
Die Tests wurden während der Entwicklung als Regressionstests verwendet, um sicherzustellen, dass Änderungen am Code keine unbeabsichtigten Seiteneffekte verursachen und alle Funktionen nach wie vor alle Anforderungen ordnungsgemäß erfüllen.

7

Evaluation

Zur Evaluation wurden während der Entwicklung des Softwaresystems mehrere Gebrauchstauglichkeits-Feldtests von verschiedenen funktionsfähigen Prototyp-Versionen durchgeführt. Dabei wurden zwei aufeinander aufbauende Versionen der App in realen Nutzungskontexten von verschiedenen Mitarbeitern der Hauptnutzergruppe, den Datenerfassern, verwendet und dies beobachtet sowie analysiert. Zusätzlich wurde auch Nutzungsfeedback von einer Mitarbeiterin der Projektbearbeiter-Nutzergruppe eingeholt, um die Nutzung in diesem Nutzungskontext zu evaluieren. Darüber hinaus fand ein Experten-Review und eine heuristische Evaluation eines Prototyps statt.
Diese nutzerzentrierte formative Evaluation ermöglicht, qualitative Daten über das Nutzungsverhalten zu sammeln und die Softwarelösung schrittweise zu verbessern und weiterzuentwickeln. Durch diesen Ansatz wurde sichergestellt, dass die Nutzer mit der App ihre Ziele effektiv, effizient und zufriedenstellend erreichen und somit in den verschiedenen Nutzungskontexten eine hohe Gebrauchstauglichkeit gewährleistet wird.

7.1 Nutzungstests des ersten Prototyps

7.1.1 Entwicklungsstand

Die ersten Nutzungstests wurden mit einem Prototyp des Systems durchgeführt, der über eine Reihe von grundlegenden Funktionen, die für die technische Gebäudebestandsaufnahme notwendig sind, verfügte. Die erste Version war wie auch die zukünftige finale Lösung als Web-App und iOS-App funktionsfähig.
Zunächst verfügte der Prototyp über die Nutzerverwaltungsfunktion für Administratoren sowie die Nutzerauthentifizierung. Ebenso war die bereits beschriebene Projektverwaltung implementiert. Abbildung 7.1 zeigt exemplarisch die Projektdetailseite mit Beispieldaten und der Möglichkeit über einen Button zu der Funktion zur Bestandsaufnahme zu gelangen.
Die Funktion der Kundenverwaltung, Fotodokumentation und Kundenanforderungen waren nicht im ersten Prototyp enthalten, da der Schwerpunkt bewusst auf die Hauptfunktion der Bestandsaufnahmeformulare gelegt wurde, da sie die Kernfunktion der App darstellen. Dazu wurden wie im Konzept beschrieben eine Seite

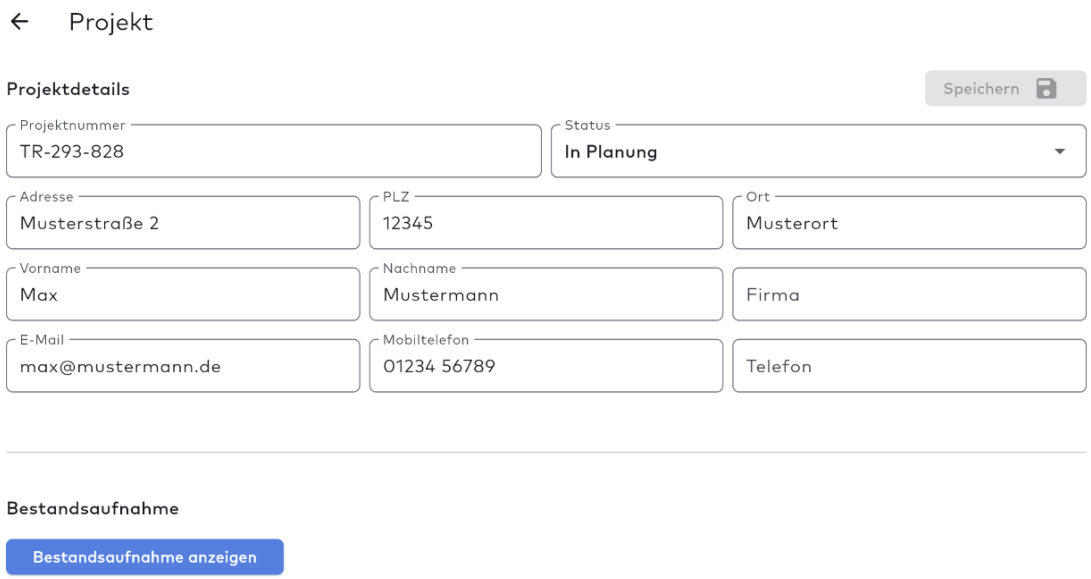


Abbildung 7.1: Screenshot der Projektdetailseite

für die Bestandsaufnahme erstellt, die am linken Bildschirmrand in einer Seitenleiste mehrere Kategorien anzeigt, die jeweils ein Formular festen Schemas und ein Freitextfeld für zusätzliche Notizen beinhalten (siehe Abbildung 7.2). Bei den Fenstern gab es wie in Abbildung 7.3 zu erkennen zusätzlich die Möglichkeit pro Fenstertyp eine eigenständige Formulargruppe zu erstellen und auszufüllen, sodass dieses Formular aus einer Liste von Fenstertyp-Formulargruppen und einem Feld für zusätzliche Notizen besteht.
Eine weitere Funktion im Rahmen der Datenerfassung war die Handschrifterkennung, die es ermöglichte, handschriftliche Notizen direkt in die Formularfelder einzugeben, die dann in Text umgewandelt wurden.
Da die Offlinefähigkeit eine zentrale Anforderung der App ist, verfügte der Prototyp auch über diese Funktion. Alle erfassten Daten wurden zunächst in einer lokalen Dankbank auf dem Endgerät gespeichert. Sobald eine aktive Internetverbindung hergestellt war, synchronisierte die App die Daten automatisch mit der Cloud-Datenbank. Der Synchronisierungsstatus konnte wie im Konzept beschrieben über eine dafür vorgesehene Seite überprüft werden.

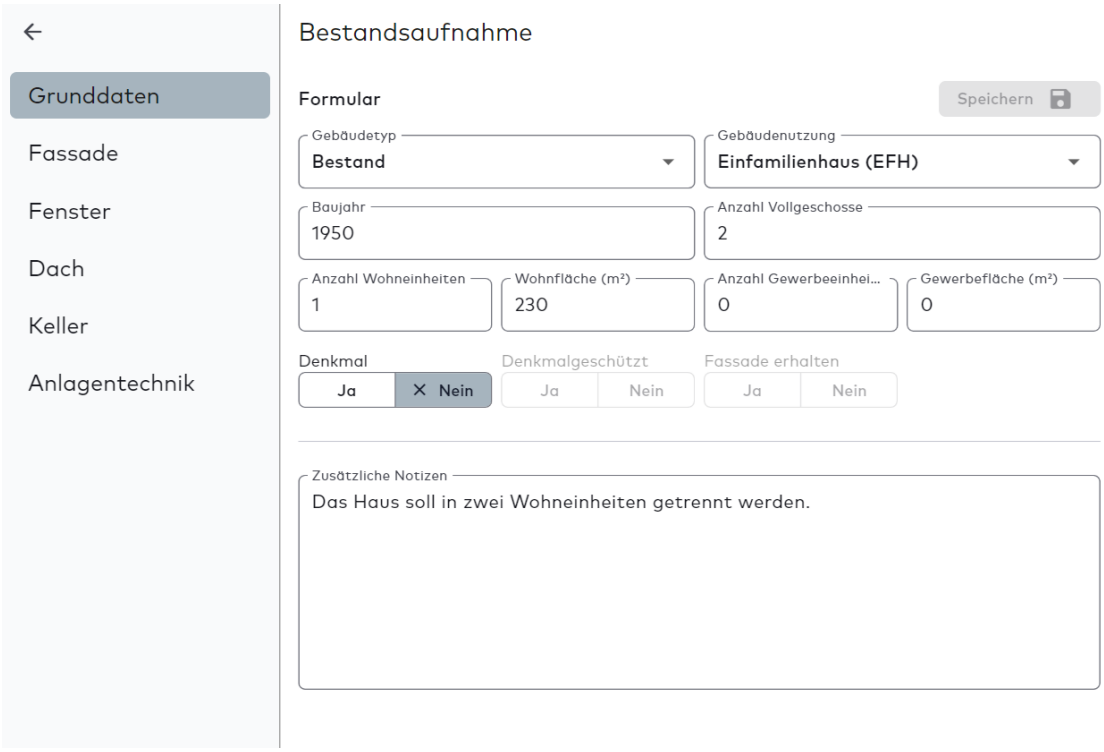


Abbildung 7.2: Screenshot des Bestandsaufnahmeformulars zu den Grunddaten des Gebäudes

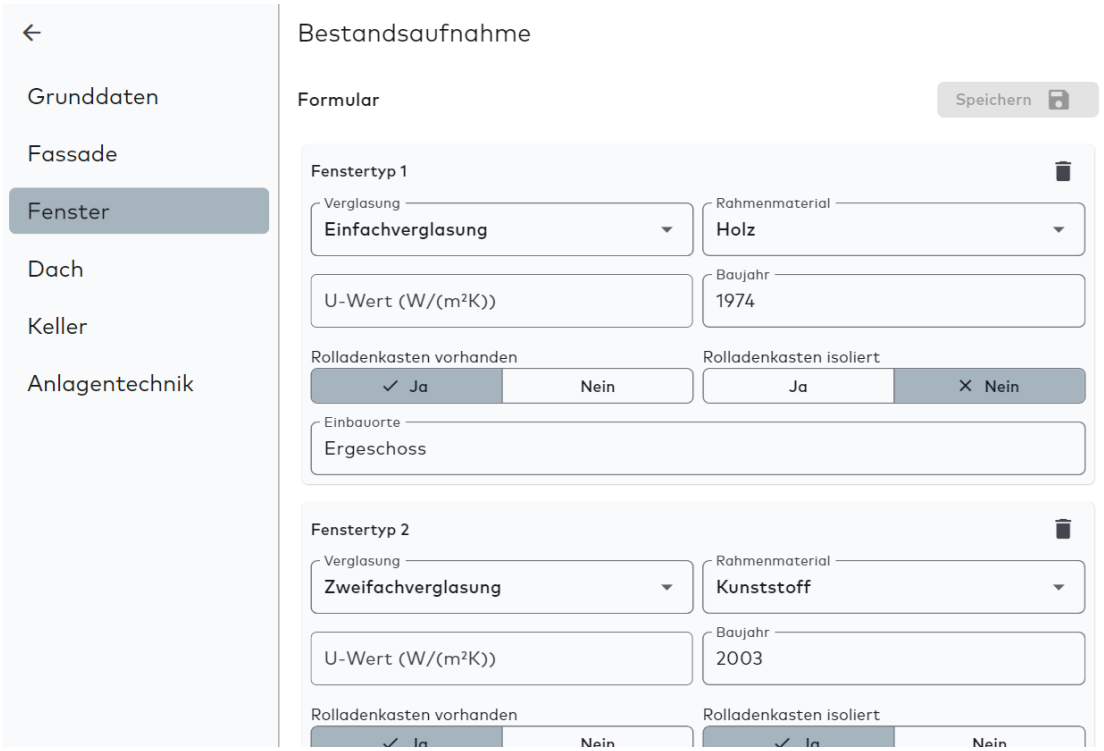


Abbildung 7.3: Screenshot des Bestandsaufnahmeformulars zu den Fenstern

7.1.2 Ziele

Im Rahmen des ersten Nutzungstests sollten mehrere spezifische Funktionen und Gebrauchstauglichkeitsaspekte der App geprüft werden, um die grundlegende Funktionalität und Nutzerakzeptanz zu bewerten. Die folgenden Ziele wurden dabei fokussiert:
Ein Ziel war es, die Sinnhaftigkeit der Kategorien, die in der App zur Bestandsaufnahme verwendet werden, zu überprüfen. Es sollte festgestellt werden, ob die Kategorien logisch aufgeteilt sind und ob sie den Anforderungen der Nutzer bei den Vor-Ort-Terminen zur Datenerfassung und bei der Projektbearbeitung im Büro gerecht werden.
Des Weiteren sollte ermittelt werden, ob die vorhandenen Formularfelder zutreffend und vollständig sind oder ob wichtige Felder fehlen. Dies sollte sicherstellen, dass die Datenerfasser alle notwendigen Informationen ohne Hindernisse erfassen können und die Projektbearbeiter alle nötigen Daten haben, um den iSFP zu erstellen.
Ein spezifischer Testbereich für die Vor-Ort-Termine durch die Datenerfasser war die Handschrifterkennung und -umwandlung. Es sollte herausgefunden werden, wie effizient die Daten per Handschrift eingegeben werden können und wie zuverlässig diese in Textform umgewandelt werden. Außerdem sollte die Akzeptanz dieser Methode bei den Nutzern ermittelt werden. Dies war entscheidend, um zu bewerten, ob die handschriftliche Eingabe als praktikable und bevorzugte Methode zur Datenerfassung genutzt werden kann.
Schließlich sollte die Synchronisationsfunktion der App im Live-Betrieb getestet werden. Dabei sollte herausgefunden werden, ob die lokal gespeicherten Daten erfolgreich synchronisiert werden, sobald eine Internetverbindung zur Verfügung steht. Diese Funktion ist besonders wichtig, um die Datenintegrität in einem realen Arbeitsumfeld sicherzustellen.
Spezifisch für die Projektbearbeiter sollte geprüft werden, ob die Software für die Erstellung eines iSFP nach dem Vor-Ort-Termin praktikabel ist und ob die Verwendung der Web-App zur Ansicht der erfassten Gebäudedaten den Prozess beschleunigt oder verlangsamt.

7.1.3 Durchführung

Die Nutzungstests wurden für die beiden Nutzergruppen, der Datenerfasser und Projektbearbeiter durchgeführt. Da sich die Nutzungskontexte stark im Ablauf und der Umgebung unterscheiden, werden diese im Folgenden getrennt voneinander betrachtet.

Datenerfasser

Im Rahmen des ersten Nutzungstests wurden drei Feldtests in realen Nutzungskontexten bei Vor-Ort-Terminen zur Datenerfassung durchgeführt. Die Benutzer waren drei verschiedene Energieeffizienzexperten aus der Hauptnutzergruppe der Datenerfasser. Diese führen regelmäßig Vor-Ort-Bestandsaufnahmen durch und

sind daher routiniert in den Arbeitsabläufen. Die Umgebung war im ersten und dritten Test neben dem Firmenbüro ein Mehrfamilienhaus und im zweiten Test ein Einfamilienhaus. Ziel war es, den Prototyp der App vor Ort für eine Gebäudebestandsaufnahme zur späteren Erstellung eines iSFP zu nutzen. Zu den Ressourcen gehörten die Web-App am PC im Büro sowie die iOS-App auf dem iPad der Mitarbeiter inkl. Apple Pencil.
Die Testpersonen erhielten eine bewusst kurzgehaltene Einführung in die App, auch um zu prüfen, ob die App weitestgehend selbsterklärend ist. Der Prototyp der App wurde zuvor auf einem iPad installiert, und die Mitarbeiter erhielten individuelle Benutzerkonten.
Während dieser Vor-Ort-Termine wurden die Mitarbeiter bei der Nutzung der App begleitet und beobachtet. Die Tests begannen im Büro, wo die Mitarbeiter mithilfe der Web-App am Computer ein neues Projekt erstellten. Daraufhin wurde das erstellte Projekt in der iOS-App geöffnet und das iPad ab diesem Zeitpunkt offline weiterbenutzt, so wie es auch im späteren Praxiseinsatz erfolgen wird. Der darauffolgende Vor-Ort-Termin bestand aus zwei Teilen, einem Gespräch am Tisch zwischen dem Energieberater und den Eigentümern des Gebäudes, indem es vor allem um die Kundenwünsche und Fragen zum iSFP ging und dem Rundgang durch die Etagen und Räume des Gebäudes. Die Reihenfolge zwischen Gespräch und Rundgang variierte je nach Termin.
Ebenso war die Reihenfolge, in der die Bestandsaufnahmekategorien beim Rundgang abgearbeitet wurden, bei den Terminen unterschiedlich, was aber durch die Navigation über das Seitenmenü kein Problem darstellte. Da die Fotodokumentation als Funktion erst in einer zukünftigen Version der App integriert wird, wurden Fotos, beispielsweise von Typenschildern, mit dem Smartphone fotografiert. Die Formularfelder der App wurden größtenteils vollständig ausgefüllt, und die Nutzung des Freitextfelds für zusätzliche Notizen war bei manchen Kategorien erforderlich. Dort wurden beispielsweise spezifische Kundenwünsche oder weitere Details eingetragen.
Die Formularfelder wurden größtenteils komplett über die Handschrifterkennung ausgefüllt, die die Wörter oder Zahlen auch in den meisten Fällen korrekt erkannte. Gelegentlich wurde die Tastatur für Eingaben oder Korrekturen dieser verwendet. Bei den Zahlenfeldern gab es Situationen, in denen die Nutzer versuchten auch textuelle Angaben wie beispielsweise „ca.“ oder mehrere Jahreszahlen einzugeben. Nach einer kurzen Erklärung, dass es sich dabei um reine Zahlenfelder handelt und für Bemerkungen das eigenständige Formularfeld genutzt werden soll, gab es damit keine Probleme mehr.
Die Formulare wurden regelmäßig über den Speichern-Button gesichert. Die App wurde beim Rundgang hauptsächlich im Hochformatmodus verwendet, während das iPad mit einer Hand gehalten und mit der anderen Hand mithilfe des Apple Pencils darauf geschrieben wurde.
Insgesamt liefen die Vor-Ort-Termine mit der App ohne technische Probleme ab und die Nutzer hatten während der Erfassung kaum Fragen oder Schwierigkeiten. Die Dauer der Vor-Ort-Termine lag dabei zwischen 45 Minuten und 1,5 Stunden.

Nach dem Termin mit dem Kunden vor Ort wurde im Büro eine Durchsicht der erfassten Daten und ein Feedback-Gespräch mit den Testnutzern durchgeführt.
Bei der Durchsicht wurde auch die Synchronisierungsfunktion getestet, wobei es zu Verständnisschwierigkeiten kam. Wie im Konzept beschrieben wurde auf der Startseite ein Button hinzugefügt, der zu einer Seite führte, auf der der Synchronisierungsfortschritt in Form einer Liste aller noch ausstehender nicht synchronisierter Änderungen verfolgt werden kann. Sobald ein Datensatz erfolgreich synchronisiert wurde, verschwand er aus dieser Liste, wodurch die Liste kürzer wurde. Die Mitarbeiter hatten Schwierigkeiten zu erkennen, ob die Synchronisierung aktuell aktiv war und durchgeführt wird. Zudem wurde nachträglich festgestellt, dass die Daten nicht erfolgreich synchronisiert wurden, da der beim Speichern der Formulardaten lokal gespeicherte Access Token zum Zeitpunkt der Synchronisierung, mehr als eine Stunde später, abgelaufen war. Dies führte dazu, dass die Cloud-Datenbank die Anfragen zur Datenspeicherung ablehnte. Da die App die Daten jedoch immer zunächst lokal gespeichert werden, gingen keine Informationen verloren und konnten zu einem späteren Zeitpunkt erfolgreich synchronisiert werden.

Projektbearbeiter

Bei der Nutzergruppe der Projektbearbeiter musste zunächst abgewartet werden, bei welchen der drei Projekte, für die mit dem Prototyp der App im Rahmen des Vor-Ort-Nutzungstests Bestandsaufnahmen durchgeführt wurden, der Praxispartner den Auftrag vom Kunden erhält. Erst wenn der Projektstatus die Angebotsphase („In Planung“) verlässt und in den Status „In Bearbeitung“ wechselt, wird beim Praxispartner von einem Projektbearbeiter ein iSFP erstellt. Dies war bis zum Abschluss dieser Arbeit nur bei einem der drei Projekte der Fall.
Da die Erstellung des iSFP ungefähr einen gesamten Arbeitstag an Zeit beansprucht, wurde die Bearbeitung nicht wie bei den Nutzungstests der Datenerfasser komplett begleitet, sondern stattdessen ein Feedbackgespräch nach der Bearbeitung mit der Mitarbeiterin durchgeführt. Die Mitarbeiterin erstellt regelmäßig Sanierungsfahrpläne vom Büro aus, bei denen die Daten von anderen Mitarbeitern vor Ort erfasst wurden, und ist daher vertraut mit dem Prozess.
Zunächst wurde dazu ein kurzes Übernahmegespräch mit dem Mitarbeiter der Datenerfassung durchgeführt. Anschließend wurde das Projekt in der Web-App geöffnet und mit der Energieberatersoftware „ZUB Helena“ der Sanierungsfahrplan erstellt.

7.1.4 Feedbackgespräch

Abschließend zu den Nutzungstests wurde mit jedem der Testpersonen ein Feedbackgespräch durchgeführt, um Rückmeldung zur Nutzung und Gebrauchstauglichkeit zu erhalten und den Nutzern die Möglichkeit zu geben, sonstige Auffälligkeiten oder Anmerkungen mitzuteilen.

Datenerfasser

Die Datenerfasser wurden zunächst danach befragt, wie sie den grundsätzlichen Aufbau und die Gliederung der Formulare in die Kategorien fanden. Das Feedback der Tester ergab, dass sie die Strukturierung der Formularfelder in Form der Kategorien, die immer im linken Seitenmenü sichtbar sind, als sinnvoll und passend empfanden. Besonders zugesagt hat ihnen, dass dies ermöglichte, die einzelnen Gebäudeteile Schritt für Schritt abzuarbeiten und mit dem Kunden durchzugehen, was dem Prozess mehr Struktur gibt als beim analogen Erfassen aller Daten in einem PDF.
Zur GUI wurde bemängelt, dass die Formularfelder bei Formulargruppen mit vielen nebeneinander angeordneten Feldern im Hochformatmodus zu schmal dargestellt werden.
Zu den Formularfeldern und der Eingabe haben die Nutzer angemerkt, dass vor allem die Dropdown-Menüs mit den Auswahlmöglichkeiten sehr hilfreich sind und die Schnelligkeit der Datenerfassung erhöhen. Insgesamt waren die Anzahl der Formularfelder ausreichend, es gab allerdings Vorschläge für weitere Felder wie die Mauerstärke der Fassade.
Dann wurden die Nutzer gefragt, wie praktikabel sie die Handschrifterkennung fanden. Dabei kam heraus, dass diese trotz anfänglicher Skepsis zufriedenstellend funktionierte. Positiv bewertet wurde, dass die geschriebenen Wörter einzeln erkannt und direkt umgewandelt werden, wodurch ein schnelles Feedback sichtbar ist. Insbesondere für das Formularfeld der zusätzlichen Notizen empfanden die Nutzer die Handschrifteingabe als sehr praktisch. Es wurde angemerkt, dass die Eingabe im ersten Moment länger dauert als eine reine Handschrifteingabe ohne Umwandlung, jedoch ist das für die Nutzer nicht störend gewesen, vor allem vor dem Hintergrund, dass dadurch das Abtippen der Daten im Büro entfällt.
Es wurde auch angemerkt, dass die Datenerfassung an einigen Stellen geringfügig länger dauert, zum Beispiel bei den Fenstern. In der Vergangenheit wurden hier meist nur Bilder gemacht und kaum weitere Daten erfasst, da dies im Rahmen der Nachbereitung geschah. Vorteil dieser Methode ist jedoch, dass die Fensterdaten nach dem Ortstermin direkt in einer strukturierten Form vorliegen und nicht erst die Bilder in der Nachbereitung durchgegangen werden müssen, um die verschiedenen Fenstertypen und deren Daten herauszuarbeiten.

Projektbearbeiter

Im Rahmen des Feedbackgesprächs wurde die Nutzerin zunächst befragt, wie sinnvoll sie die Aufteilung der Kategorien der Bestandsaufnahme für den Arbeitsprozess empfunden hat. Sie merkte an, dass die Reihenfolge nicht komplett zu dem Vorgehen bei der Erstellung eines iSFP passt, aber dies kein Problem darstellt, da nach Belieben zwischen den verschiedenen Kategorien über das Seitenmenü zugegriffen werden kann. Die Aufteilung wurde von ihr zudem als sinnvoll und passend empfunden.
Zur Vollständigkeit der Formulare wurde rückgemeldet, dass in manchen der Formulare noch einige für den Sanierungsfahrplan relevante Felder fehlen, darunter

beispielsweise Angaben darüber, wie der Kellerabgang bei unbeheizten Kellern gedämmt ist und ob es sich bei dem Gebäude um eine Grenzbebauung handelt.
Zum Zeitaufwand berichtete die Mitarbeiterin, dass sie bei der Bearbeitung eines Projekts mit der App schneller sei als mithilfe der PDF-Bestandsaufnahmetabelle, weil mehr Informationen zum Gebäude über die App dokumentiert werden können. Dadurch, dass die Bearbeiterin so mehr Daten und Informationen zum Gebäude gebündelt einsehen konnte, musste sie weniger Pläne oder Bilder hinzuziehen, um nötige Informationen für den iSFP herauszufinden, was den Arbeitsablauf beschleunigt.

7.1.5 Erkenntnisse und Maßnahmen

Aus den Nutzungstests haben sich eine Reihe von Erkenntnissen ergeben, die zu Maßnahmen bei der weiteren Entwicklung geführt haben.
Bei der Projekterstellung wurde deutlich, dass es Fälle gibt, in denen der Besitzer des Gebäudes nicht in diesem wohnt, sondern das Gebäude vermietet. Dadurch entsteht die Notwendigkeit, sowohl die Adresse des Kunden als auch die des Gebäudes zu berücksichtigen. In der nächsten Version der App wird dieses Problem durch die konzipierte Kundenverwaltung gelöst, bei der jeder Kunde mit einer Adresse abgespeichert und einem Projekt mit einer bei Bedarf abweichenden Adresse zugeordnet wird.
Die Strukturierung der Kategorien für die Bestandsaufnahme erwies sich als hilfreich und angemessen für den Arbeitsablauf, indem sie eine strukturierte Durchführung aller Aufgaben ermöglichte. Dies stellt eine klare Verbesserung gegenüber dem Arbeitsablauf ohne App dar.
Die Annahme, dass die Bestandsaufnahme-Seiten hauptsächlich im Querformat genutzt werden würden, erwies sich als falsch, da es sich laut Testern beim Vor-Ort-Termin als praktischer erwies, das iPad im Hochformat mit einer Hand zu halten und mit der anderen Hand darauf zu schreiben. Trotzdem ist auch die Querformat-Ansicht wichtig, da diese vor allem dann zum Einsatz kommt, wenn die Mitarbeiter mit den Kunden am Tisch sitzen und das Tablet dabei mit der Hülle aufgestellt haben, um die Besprechung zu erleichtern. Aufgrund dieser Erkenntnis wird in der nächsten Version der App sichergestellt, dass sich das Seitenmenü einklappen lässt, sodass alle Formularfelder auch bei einer geringeren Bildschirmbreite angemessen groß und damit effektiv bedienbar sein.
Eine weitere Erkenntnis ist, dass bei manchen Formularkategorien weitere Felder, wie beispielsweise ein Feld zur Angabe der Wandstärke, ergänzt werden müssen und vorhandene Felder wie das Feld für den Fenster-Lambda-Wert entfernt werden sollen, da diese Angaben in der Regel nicht beim Vor-Ort-Termin ermittelt werden können.
Die Seite zum Synchronisationsstatus erwies sich als verbesserungswürdig, da der aktuelle Status der Synchronisation für die Nutzer nicht klar erkennbar war. Ebenfalls wird in der nächsten Version das Problem mit dem abgelaufenen Access-Token behoben.
Ein weiteres identifiziertes Problem war der Standardfilter („in Bearbeitung“) in

der Projektübersicht, der bei den Mitarbeitern Verwirrung auslöste, da neue Projekte erst nach Änderung des Filters sichtbar wurden. Dieser Prozess sollte so optimiert werden, dass standardmäßig alle Projekte dargestellt und die Filter optional angewendet werden können.
Die Handschrifterkennung stellte sich als gewöhnungsbedürftig, aber effektiv heraus, vor allem weil die Notizen nicht mehr nachträglich abgetippt werden müssen. Die Mitarbeiter müssen sich zunächst mit dieser Funktion vertraut machen, bevor sie effizient genutzt werden kann.
Es wurde außerdem bemerkt, dass es ohne ein zurück navigieren zur Projektseite nicht ersichtlich ist, in welchem Projekt sich der Benutzer aktuell befindet, während die Bestandsaufnahme erstellt wird. Diese fehlende Anzeige der Projektnummer oder des Projektnamens wird in der nächsten Version hinzugefügt, um die Übersichtlichkeit zu verbessern.
Aus jeder der genannten Erkenntnisse wurden Maßnahmen für die weitere Entwicklung erstellt, um sicherzustellen, dass die App in der Praxis effektiv, effizient und zufriedenstellend genutzt werden kann und somit eine hohe Gebrauchstauglichkeit aufweist.

7.1.6 Fazit

Die Evaluation des ersten Prototyps hat wertvolle Einblicke in die Nutzung und Gebrauchstauglichkeit der App geliefert. Die Tests haben gezeigt, dass die analysierten Annahmen und daraus resultierenden Nutzungsanforderungen, die bei der Konzeptionierung und Implementierung berücksichtigt wurden, weitestgehend zutreffen.
Die Vor-Ort-Tests zur Gebäudebestandsaufnahme haben ergeben, dass die Strukturierung der Formulare in Kategorien sinnvoll und zutreffend ist und dem Arbeitsablauf zusätzliche Struktur verleiht. Bei der Datenerfassung durch die Formulare erwiesen sich vor allem die Dropdown-Auswahlfelder als effizienzsteigernd. Die spezifische Funktion der Handschrifterkennung, wurden trotz anfänglicher Skepsis positiv aufgenommen und als praktisch bewertet, besonders für die Erfassung von Notizen. Es wurden jedoch auch Bereiche identifiziert, in denen Verbesserungen notwendig sind, wie die Anpassung und Erweiterung der Formularfelder, Anpassungen an der Seitenleiste sowie die Optimierung und Fehlerbehebung bei der Synchronisationsfunktion.
Im Vergleich zum bisherigen Ablauf ohne App wurde festgestellt, dass die Datenerfassung vor Ort durch die Handschrifterkennung und die zusätzliche textuelle Datenerfassung der Fenstertypen geringfügig länger dauert, dies aber durch eine verringerte Nachbereitungszeit kompensiert wird.
Die Nutzung der Web-App bei der Projektbearbeitung im Büro hat gezeigt, dass die Aufteilung der Formulare für die Bestandsaufnahme besonders durch die leichte Navigation über das Seitenmenü auch für diesen Nutzungskontext geeignet ist. Bei einigen Formularen wurde festgestellt, dass für die Erstellung des iSFP relevante Felder fehlen. Insgesamt hat sich jedoch herausgestellt, dass die App die Bearbeitung effizienter gestaltet, da mehr Gebäudedaten erfasst und einsehbar sind, was

die Notwendigkeit für das das Hinzuziehen von Plänen und Fotos verringert und somit den Arbeitsablauf beschleunigt.
Zusammenfassend kann gesagt werden, dass der erste Prototyp eine solide Basis für die weitere Entwicklung darstellt. Die Erkenntnisse aus den Nutzungstests haben direkt zu konkreten Maßnahmen für die weitere Entwicklung geführt, um die App weiter zu optimieren und ihre Gebrauchstauglichkeit zu erhöhen.

7.2 Expertenfeedback und heuristische Evaluation

7.2.1 Expertenfeedback

Nach der Implementierung der Maßnahmen, die sich aus den Erkenntnissen der ersten Nutzungstests ergeben haben, wurde der Stand der App, insbesondere die Bestandsaufnahmeformulare, mit dem Praxispartner zusammen durchgegangen, um Expertenfeedback zu erhalten. Dazu wurde zuvor eine Liste mit allen von den Nutzern beim ersten Nutzungstest vorgeschlagenen fehlenden Formularfeldern erstellt und zusammen mit dem Praxispartner ein Konzept zur Eingliederung dieser in die bestehenden Formulare erstellt. Neben den von den Nutzern vorgeschlagenen Feldern wurden auch vom Praxispartner weitere fehlende Felder ergänzt, die wichtig für eine vollständige Datenerfassung sind.
Durch diese Ergänzungen sind die Formulare im Umfang gewachsen, wodurch sie unübersichtlich wurden. Dieses Problem wurde behoben, indem die Formularfelder in benannte Abschnitte unterteilt wurden, damit das Formular mehr Struktur hat und klar ist, wozu ein Formularfeld gehört. Abbildung 7.4 zeigt exemplarisch das Formular zu den Gebäudegrunddaten mit den neuen benannten Formularabschnitten.
Zusätzlich wurde darauf hingewiesen, dass der Titel der Formularkategorie „Fassade“ in diesem Kontext nicht korrekt ist und durch den Begriff „Außenwand“ ersetzt werden muss. Auch Formulierungen an anderen Stellen wurden korrigiert, damit die App die Fachsprache der Benutzer verwendet.

7.2.2 Heuristische Evaluation

Bei einer heuristischen Evaluation ist aufgefallen, dass der Speichern-Button nicht sichtbar ist, wenn auf den Formularseiten ganz nach unten gescrollt wird, da sich der Button oberhalb des Formulars aber im Inhaltsbereich der Seite befindet. Um sicherzustellen, dass der Button auch beim scrollen nach unten immer sichtbar bleibt und somit die Nutzer nicht vergessen, ihre Änderungen zu speichern, wurde der Speichern-Button in die obere fixierte Menüleiste verschoben.
Zudem ist aufgefallen, dass es bei den Dropdown-Menüs der Formulare keine Möglichkeit gibt etwas auszuwählen, wenn die angegebenen Auswahlen nicht zutreffen. Somit würde der Nutzer nichts auswählen, wenn beispielsweise die Bauweise der Außenwand von den Vorauswahlen abweicht, was darauf hinweisen könnte, dass die Bauweise unbekannt ist. Um dies zu verhindern und sicherzustellen, dass der

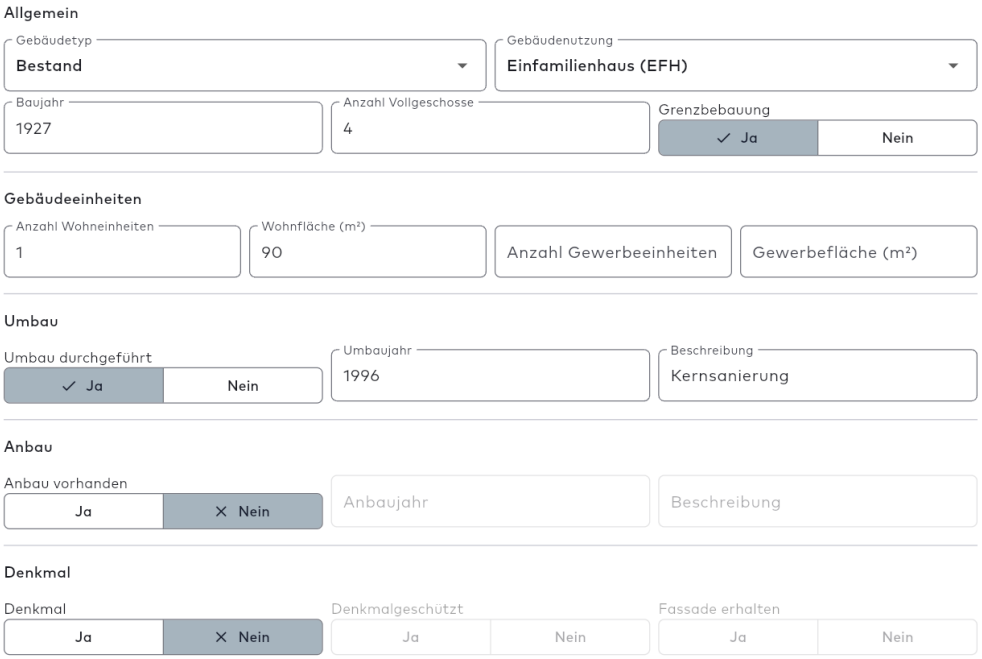


Abbildung 7.4: Screenshot des Formulars zu den Grunddaten mit den neuen benannten Formularabschnitten

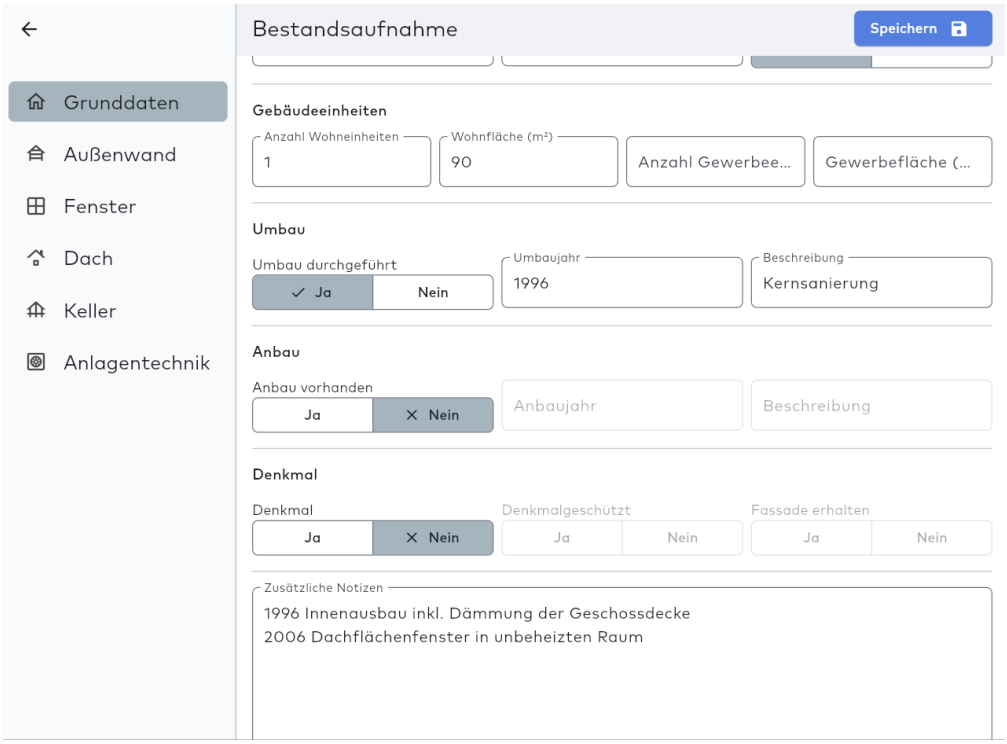


Abbildung 7.5: Screenshot eines heruntergescrollten Formulars mit neuer Platzierung des Speichern-Buttons in der oberen Menüleiste

Nutzer auch in solchen Fällen eine Auswahl treffen kann, enthalten die Dropdown-Menüs der Formulare nun immer eine „Andere“-Option. Dies ermöglicht es den Nutzern, eine alternative Auswahl zu treffen, falls keine der vorgegebenen Optionen zutrifft. Ein zusätzlicher Vorteil dieser Anpassung ist die Möglichkeit, nach einigen Monaten der Datenerfassung zu analysieren, bei welchen Feldern häufig „Andere“ gewählt wird. Basierend auf diesen Erkenntnissen können dann bei Bedarf spezifischere Optionen ergänzt werden.
Bei der Entwicklung und Weiterentwicklung der App ist es unvermeidbar, dass sich das zugrundeliegende Schema der Datenbank ändert, wenn beispielsweise neue Felder hinzukommen oder andere entfernt werden. In solchen Fällen sind alte Versionen der App nicht mehr mit der Cloud-Datenbank kompatibel, was zu Fehlern beim Speichern oder Abrufen von Daten führen kann. Um dieses Problem zu vermeiden, zeigt die App nun wie in Abbildung 7.6 zu sehen einen Hinweisbanner auf der Startseite an, wenn die aktuelle installierte Version veraltet und nicht mehr mit der Cloud-Datenbank kompatibel ist.

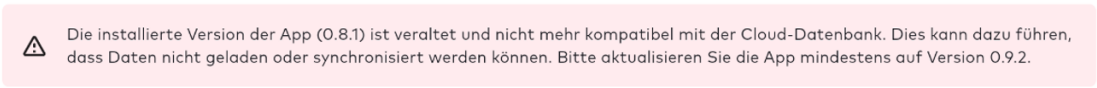


Abbildung 7.6: Screenshot des App-Versionshinweis-Banners auf der Startseite

7.3 Nutzungstests des zweiten Prototyps

7.3.1 Entwicklungsstand

Die größte Neuerung des zweiten Prototyps im Vergleich zum ersten ist die Kundenverwaltung. Neben der Projektverwaltung gibt es nun wie in Abbildung 7.7 zu sehen eine Kundenverwaltung, wodurch eine Verknüpfung zwischen Kunden und Projekten möglich ist. Kunden verfügen nun über einen Namen, eine Nummer, einen Typ und eine Adresse. Ein Projekt kann entweder die Adresse des Kunden oder eine abweichende Projektadresse haben. Zudem wurde bei den Kundendaten ein Feld für interne Notizen hinzugefügt, da dies für den Auftraggeber wichtig war.

Eine weitere bedeutende Neuerung ist die in Abbildung 7.8 dargestellte Funktion zur Erfassung von projektspezifischen Kundenanforderungen. Dazu wurden drei verschiedene neue Formulare erstellt: ein Formular zur Erfassung der aktuellen und zukünftigen Bewohner, ein Formular zur Erfassung der Kundenwünsche und ein Formular zur Aufnahme des Budgets und Zeitplans des Projekts. Für diese neuen Seiten wurde das gleiche Layout und Design wie für die Bestandsaufnahme verwendet.
Um den Nutzern die Möglichkeit zu bieten, schnell zu den zuletzt bearbeiteten Projekten zu navigieren, wurde wie im Konzept beschrieben auf der Startseite ein Bereich integriert, der die zuletzt erstellten Projekte eines Nutzers anzeigt. Ein

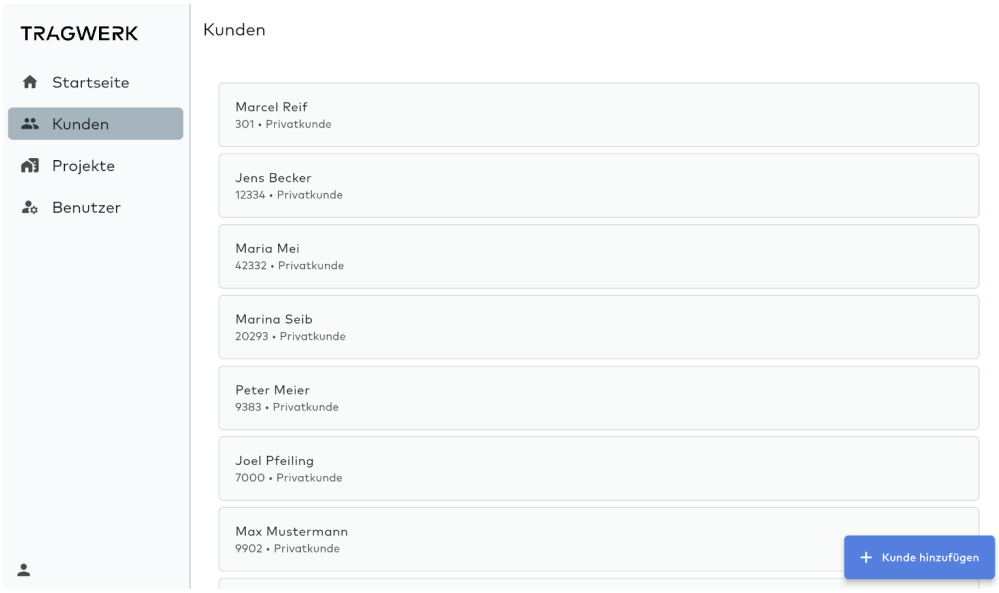


Abbildung 7.7: Screenshot der Kundenverwaltung

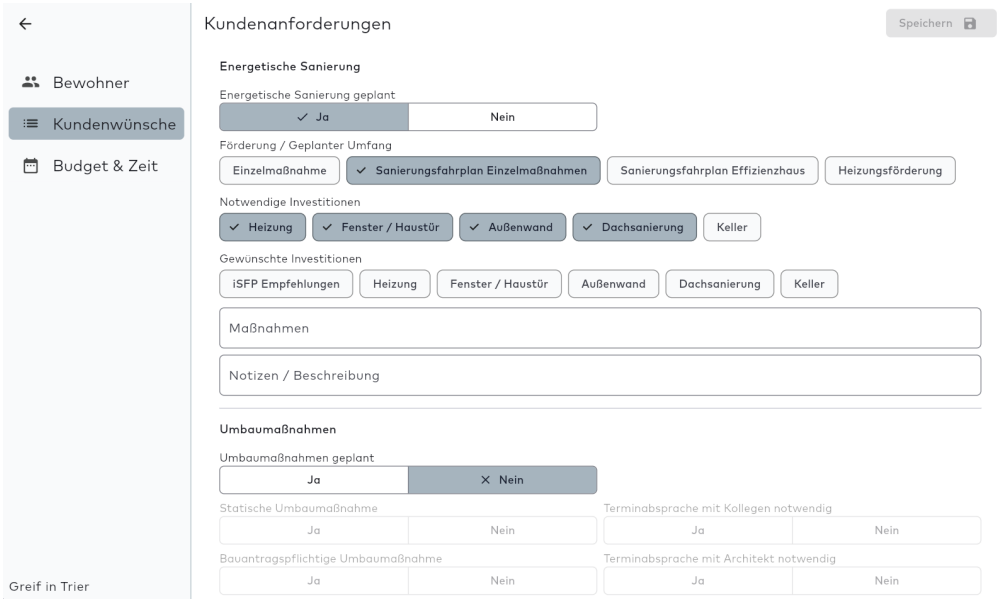


Abbildung 7.8: Screenshot des Kundenwünsche-Formulars der Kundenanforderungen

Screenshot dazu ist in Abbildung 7.9 zu sehen.
Außerdem wurde das Seitenmenü für die Basisseiten wie im Konzept beschrieben implementiert. Dies ermöglicht eine einfachere Navigation zwischen Kunden, Projekten und weiteren Seiten der App. Bei der Navigation zu den Bestandsaufnahmeseiten, die ihre eigene Seitenleiste haben, wird das Seitenmenü der Basisseiten ausgeblendet, um nicht zu viel horizontalen Platz des Bildschirms einzunehmen.
Die aus dem ersten Test resultierende Maßnahme, die Bestandsaufnahmeseite bes-

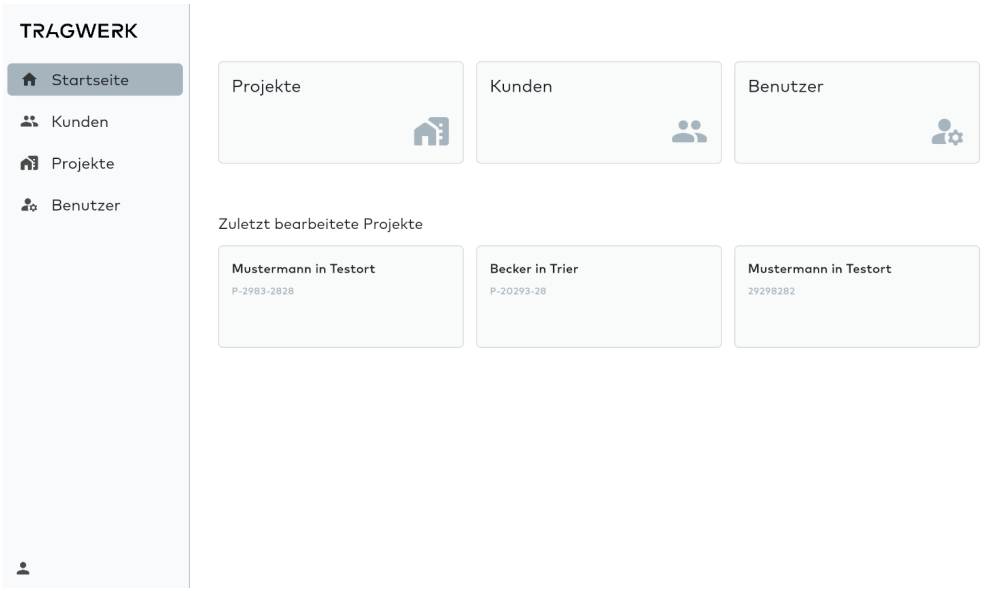


Abbildung 7.9: Screenshot der Startseite

ser für die Benutzung im Hochformatmodus anzupassen, wurde so umgesetzt, dass das Seitenmenü nun wie in 7.10 zu sehen einklappbar ist und anstelle des Namens des Formulars ein Icon anzeigt, wodurch mehr Platz für das Formular bleibt. Wenn die App im Hochformatmodus benutzt wird, ist das Seitenmenü standardmäßig eingeklappt. Die Icons im eingeklappten Zustand wurden so gewählt, dass auch ohne die Reihenfolge zu lernen sichtbar ist, um welches Formular es sich handelt. Um den Lerneffekt weiter zu verstärken, werden die Icons auch im ausgeklappten Zustand angezeigt.
Außerdem wird der Projektname des Projekts, für das die Bestandsaufnahme ausgefüllt wird, nun im ausgeklappten Zustand der Seitenleiste am unteren Ende dieser angezeigt, um die Rückmeldung aus dem ersten Nutzungstest zu adressieren.
Zudem wurde das Synchronisierungsproblem aufgrund des abgelaufenen Access Tokens (JWT) gelöst. Auf die technische Implementierung dessen wird im Implementierungskapitel genauer eingegangen. Darüber hinaus wurde die Synchronisierungsseite überarbeitet, indem jetzt angezeigt wird, ob die Synchronisierung aktuell durchgeführt wird, wodurch der Systemstatus nun einfacher zu erkennen ist.
Die Funktion der Fotodokumentation konnte aus zeitlichen Gründen nicht implementiert werden, vor allem vor dem Hintergrund, dass diese bezüglich der Speicherung, Freigabe und Synchronisierung relativ komplex ist. Diese wird in einer nächsten Version nach Abschluss dieser Arbeit ergänzt und ebenfalls nutzerzentriert evaluiert.

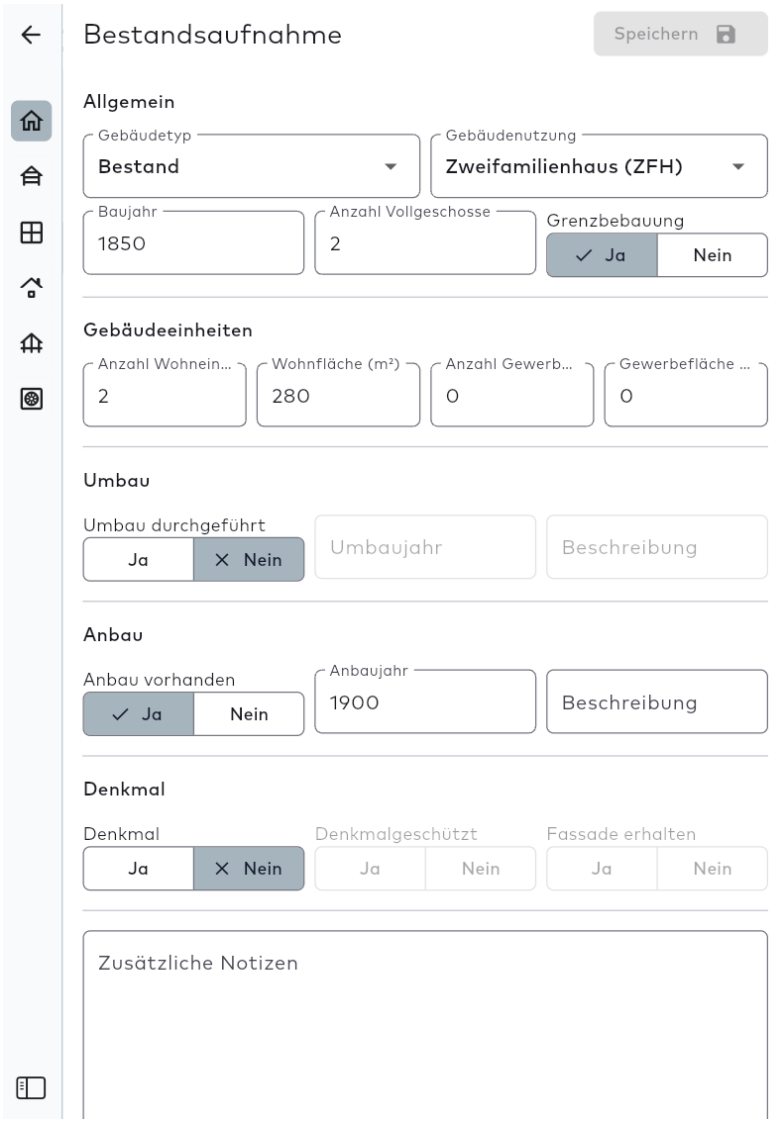


Abbildung 7.10: Screenshot der eingeklappten Seitenleiste bei der Bestandsaufnahme

7.3.2 Ziele

Im Rahmen der Nutzungstests des zweiten Prototyps soll überprüft werden, ob die neuen Funktionen der Kundenverwaltung und die Erfassung von Kundenanforderungen verständlich umgesetzt wurden und im Praxiseinsatz praktikabel nutzbar sind.
Zudem soll ermittelt werden, ob die Formulare nun alle relevanten Informationen für die Projektbearbeitung enthalten sind und ob die umfangreicheren Formulare für die Datenerfasser trotzdem übersichtlich sind.
Weiterhin soll untersucht werden, ob die Neuerung mit der einklappbaren Seitenleiste ausreicht, um breite Formulare auch im Hochformat effizient ausfüllen zu können.

Für diese Tests wird die App nicht nur auf den 10,9 Zoll großen iPads der Mitarbeiter getestet, sondern auch auf einem 13 Zoll großen iPad. Dabei soll festgestellt werden, wie sich die Displaygröße auf die Nutzung der App auswirkt und welche Größe in Zukunft vorzuziehen ist.

7.3.3 Durchführung

Datenerfasser

Der zweite Prototyp wurde erneut bei drei Vor-Ort-Bestandsaufnahmen von Mitarbeitern der Nutzergruppe der Datenerfassung in realen Nutzungskontexten bei Kunden getestet. Die Nutzungsumgebung der Tests umfasste sowohl das Firmenbüro als auch Einfamilienhäuser, von denen sich eines im Umbau befand. Die Aufgabe der Mitarbeiter bestand darin, alle projekt- und kundenbezogenen Daten zu erfassen, insbesondere die Gebäudebestandsaufnahme mit der App durchzuführen. Zu den Ressourcen gehörten wie auch beim ersten Nutzungstest die Web-App am PC im Büro sowie die iOS-App auf dem iPad der Mitarbeiter inklusive Apple Pencil. Die Mitarbeiter hatten die Gelegenheit, die App auf dem 10,9 Zoll und dem 13 Zoll großen iPad zu testen.
Der Ablauf begann mit der Erstellung des Kunden und des Projekts über die Web-App im Büro. Beim Vor-Ort-Termin wurden dann die Formulare zu den Kundenanforderungen und der technischen Gebäudebestandsaufnahme offline während des Gesprächs mit dem Kunden am Tisch und während des Rundgangs durch das Gebäude auf dem iPad ausgefüllt. Bei allen Tests wurde zunächst mit den Kundenanforderungen begonnen und anschließend die technischen Daten beim Rundgang erfasst. Die Dauer der Vor-Ort-Termine lag zwischen ein und zwei Stunden. Nach dem Kundentermin wurden die Daten im Büro synchronisiert, sodass diese zur Weiterbearbeitung am Computer bereitstanden. Bei allen Tests funktionierte die Synchronisierung reibungslos. Abschließend wurde ein Feedbackgespräch mit den Nutzern durchgeführt, um Rückmeldungen zur Nutzung zu erhalten.
Insgesamt traten während der Nutzung keine technischen Probleme oder Schwierigkeiten seitens der Nutzer auf. Alle Daten wurden erfolgreich erfasst und standen nach der Synchronisierung im Büro zur Weiterverarbeitung bereit.

Projektbearbeiter

Der zweite Prototyp der App konnte von der Nutzergruppe der Projektbearbeiter nicht für die Erstellung eins iSFP getestet werden, da zwischen den Vor-Ort-Nutzungstests des zweiten Prototyps und dem Abschluss dieser Arbeit keine Beauftragung des Praxispartners durch einen Kunden stattfand. Trotzdem wurde ein Feedbackgespräch zu den Veränderungen im zweiten Prototyp mit der Mitarbeiterin der Projektbearbeitung, die bereits einen iSFP mit dem ersten Prototyp der App erstellt hat, durchgeführt.

7.3.4 Feedbackgespräch

Datenerfasser

Im Feedbackgespräch im Anschluss an den Vor-Ort-Termin wurden die Mitarbeiter zunächst gefragt, ob die Nutzung der App zufriedenstellend war und wie ihr Eindruck dieser Version der App ist.
Der Mitarbeiter des ersten Tests dieses Prototyps äußerte, dass durch die Erweiterung der Formulare im Vergleich zum vorherigen Prototyp eine ausführlichere Einarbeitung vor dem direkten Kundentermin nötig gewesen wäre, um die Formulare besser zu verstehen und im Kundengespräch effizienter auszufüllen. Diese Rückmeldung ist verständlich und wird für die nächsten Tests mit anderen Mitarbeitern berücksichtigt. Der gleiche Mitarbeiter äußerte jedoch auch, dass er die generelle Nutzung der App als angenehm empfand und die App dem Vor-Ort-Termin einen strukturierten Ablauf verleiht.
Das Feedback hat auch ergeben, dass es durch den Umfang und die Strukturierung der Formulare unwahrscheinlicher ist, dass vor Ort vergessen wird, bestimmte Daten zu dokumentieren. Außerdem wurden erneut die Dropdown-Formularfelder mit den Auswahlmöglichkeiten als sinnvoll hervorgehoben, da diese zusätzlich zur Eingabeschnelligkeit den Vorteil haben, dass der Erfasser direkt alle gängigen Möglichkeiten vor sich sieht und diese daher nicht auswendig kennen muss.
Die neue Funktion zur Kundenverwaltung wurden von den Nutzern als positiv und verständlich bewertet. Bei der ebenfalls neuen Funktion zur Erfassung der Kundenanforderungen wurde vorgeschlagen, dass die Formulare dieser auf der gleichen Seite wie die Bestandsaufnahme sein sollten, sodass vom Seitenmenü aus sowohl auf die Formulare zu den Kundenanforderungen als auch die Formulare zur Bestandsaufnahme zugegriffen werden kann. Ein anderer Mitarbeiter empfand jedoch die Trennung dieser Funktionen als sinnvoll und nicht störend. Um dazu eine Entscheidung zu treffen, bedarf es weiterer Nutzertests zur Gegenüberstellung der Argumente. Die beiden Formulare wurden in der Entwicklung getrennt, da die Kundenanforderungen vor dem Rundgang durch das Haus separat aufgenommen werden sollten, um zu gewährleisten, dass kein Formular vergessen wird.
Zur Frage nach der Vollständigkeit wurden von den Testern einzelne fehlende Felder und Auswahlmöglichkeiten bei bestehenden Feldern angemerkt. Grundsätzlich sind die Formulare jedoch laut Mitarbeitern sehr detailliert und erfassen alle wichtigen Merkmale zum Gebäude. Trotz der umfangreichen Formulare sind diese laut Mitarbeitern dennoch übersichtlich und zufriedenstellend nutzbar.
Ein Nutzer merkte bei den Formularen zum Keller, genauer gesagt bei Abschnitten zum Kellerabgang und den Innenwänden an, dass es unklar ist, ob sich die Felder zur Dämmung auf die Tür oder den gesamten Kellerabgang beziehen. Hier sollte die Reihenfolge der Felder angepasst werden, um dies klarzustellen.
Negativ wurde von einem Mitarbeiter angemerkt, dass es bei den Zahlenfeldern nicht möglich ist, zusätzlich auch Text oder Anmerkungen einzutragen und diese dann unter dem Formular in das Freitextfeld notiert werden müssen. Hierzu ist zu sagen, dass die Erlaubnis, dort auch Bemerkungen in Form von Text einzutragen, dazu führen würde, dass die Zahlenfelder deutlich größer werden müssten

und zusätzlich das Datenmodell ungenauer wird, was es erschwert, die Daten in der Zukunft automatisiert zu analysieren.
Die einklappbare Seitenleiste wurde insbesondere auf dem 10,9 Zoll großen iPad im Hochformat als Verbesserung wahrgenommen, da so mehr Platz für Felder und deren Bezeichner bleibt. Dadurch waren auch im Hochformat alle Formulare effizient ausfüllbar.
Zur Größe des iPads äußerten alle Tester, dass das Schreiben mit dem Stift auf dem großen iPad angenehmer sei, da mehr Platz zum Schreiben und zum Auflegen des Handballens besteht. Einer der Tester merkte an, dass auf dem großen iPad viel mehr der Formularfelder gleichzeitig sichtbar sind, wodurch er sich schwerer auf die einzelnen Abschnitte konzentrieren konnte. Auch wurden Bedenken geäußert, dass das größere iPad zukünftig beim Aufnehmen von Bildern in engen Räumen Schwierigkeiten mit sich bringen könnte. Grundsätzlich zeigt sich aber, dass die App auf beiden Größen zufriedenstellend nutzbar ist und weitere Tests zeigen werden, welche Größe in Zukunft am besten geeignet ist.
Abschließend wurden die Tester befragt, wie sich die Vor-Ort-Bestandsaufnahme mit der App von dem Ablauf ohne App unterscheidet. Es wurde festgestellt, dass mit der App vor Ort deutlich mehr Daten erfasst werden, was insbesondere dann vorteilhaft ist, wenn ein anderer Mitarbeiter das Projekt weiterbearbeitet. Zudem wurde angemerkt, dass ohne die App ein ausführlicheres Kundengespräch stattgefunden hat, während nun mehr Fokus und Konzentration auf das Ausfüllen des Formulars gelegt wird. Es ist jedoch zu erwarten, dass sich dies mit zunehmender Nutzung verbessert, da die Datenerfasser die Formulare besser kennenlernen und flüssiger in das Kundengespräch integrieren können.

Projektbearbeiter

Im Rückmeldungsgespräch mit einer Projektbearbeiterin wurde zu Beginn über die neuen Formulare zu den Kundenanforderungen gesprochen. Die Mitarbeiterin erklärte, dass die Angabe zu den Bewohnern des Hauses zwar nicht notwendig für die Erstellung des iSFP sei, aber dennoch hilfreich, um einen größeren Kontext zum Projekt zu erhalten. Das Formular zu den Kundenwünschen, das unter anderem die gewünschten energetischen Maßnahmen beinhaltet, sei sehr hilfreich, da sie dadurch weiß, welche Maßnahmen in den Sanierungsfahrplan integriert werden sollen. Auch wenn einige dieser Informationen in der bisherigen PDF-Bestandsaufnahmetabelle zur Vor-Ort-Datenerfassung vorhanden waren, wurden diese laut Aussage der Mitarbeiterin in der Vergangenheit nicht immer ausgefüllt. Zusätzlich sind die Formulare in der App ausführlicher, da sie beispielsweise auch enthalten, in welchem Umfang die Sanierungen durchgeführt werden sollen, etwa ob das Ziel die Umsetzung von Einzelmaßnahmen oder die Erreichung eines Effizienzhauses ist. Ebenso sei die Erfassung des Budgets nützlich,um zu bestimmen, welche Maßnahmen in welchen Paketen vorgeschlagen werden können.
Die neuen Abschnitte, welche die Felder der Formulare gliedern, wurden als sehr hilfreich empfunden, insbesondere hinsichtlich der Übersichtlichkeit.
Zur Vollständigkeit der Formularfelder wurde angemerkt, dass nun keine für die

Bearbeitung relevanten Felder mehr fehlen.
Abschließend betonte die Mitarbeiterin, dass sie zufrieden mit der App sei und weiteres Feedback rückmelden werde, sobald ein Projekt mit dem neuesten Stand der App bearbeitet wurde.

7.3.5 Erkenntnisse und Maßnahmen

Aus den Nutzungstests haben sich einige kleinere Erkenntnisse ergeben, aus denen konkrete Maßnahmen zur weiteren Entwicklung abgeleitet wurden, die nach Abschluss dieser Arbeit im Rahmen der weiteren Zusammenarbeit mit dem Praxispartner umgesetzt werden.
Zunächst fiel auf, dass der erste Tester die Formulare nur selten speicherte. So kam es mehrfach vor, dass ihm das Pop-up „Ungespeicherte Änderungen“ angezeigt wurde, und somit verhindert wurde, dass er die Seite verlässt, bevor alle Formulare gespeichert wurden. Er musste dann alle Formulare durchgehen, um jenes zu finden, welches noch nicht gespeichert war. Dies wird für die nächsten Tests vereinfacht, indem im Pop-up ein Button integriert wird, über den alle ungespeicherten Formulare auf einmal gespeichert werden können.
Bei der Beobachtung der Nutzung wurde festgestellt, dass das Feedback-Banner das bei gewissen Aktionen kurz am unteren Bildschirmrand angezeigt wird, bei der Navigation zu Seiten, die über einen Aktionsbutton am rechten unteren Rand verfügen, diesen überdeckt. Dies führte im Test dazu, dass ein Nutzer nach der Erstellung eines Kunden den Button zur Erstellung eines Projektes erst vollständig sehen konnte, als das Banner nach einigen Sekunden wieder ausgeblendet wurde. Zukünftig wird dieses Problem dadurch gelöst, dass sich der Aktionsbutton bei Einblendung der Feedbackbanners nach oben verschiebt, sodass der Button stets sichtbar bleibt.
Die bei den Tests aufgefallenen fehlenden Formularfelder und Auswahlmöglichkeiten werden ebenfalls in der nächsten Version der App ergänzt und die unpassende Platzierung gewisser Felder wird behoben.

7.3.6 Fazit

Durch die Nutzungstests des zweiten Prototyps konnten die neuen oder veränderten Funktionen der App getestet und durch die Nutzer in verschiedenen Nutzungskontexten bewertet werden. Dabei gab es neue Feststellungen und Erkenntnisse.
Zunächst wurde festgestellt, dass die umfangreicheren Formulare trotzdem übersichtlich sind und dazu beitragen, dass vor Ort keine Angaben vergessen werden. Bei den Kundenanforderungen gab es den Vorschlag, diese in die Bestandsaufnahme zu integrieren, wobei keine einheitliche Meinung unter den Mitarbeitern bestand. Kleine Verbesserungspotenziale wurden bei den Formularfeld-Auswahlmöglichkeiten, der Anordnung mancher Formularfelder, der Formularspeicherung und dem Anzeigen von Hinweisbannern identifiziert.
Die Größe des iPads bringt Vorteile, insbesondere beim handschriftlichen Ausfüllen der Formulare, aber auch Nachteile hinsichtlich der Handlichkeit. Grundsätzlich

lässt sich die App jedoch auf beiden Größen effizient bedienen, wozu auch die Anpassungen an der Seitenleiste beigetragen haben.
Im Vergleich zum Arbeitsablauf ohne App konnten die Datenerfasser feststellen, dass mit der App vor Ort deutlich mehr Daten erfasst werden, was vorteilhaft für die Arbeitsteilung eines Projekts ist. Allerdings reduzierte die Abarbeitung der Formulare im Test den Fokus auf das Kundengespräch. Es wird jedoch erwartet, dass sich dies mit zunehmender Nutzung verbessert und die Formulare flüssiger in das Gespräch integriert werden können.
Seitens der Projektbearbeitung im Büro wurden die Formulare zu den Kundenanforderungen als sehr nützlich angesehen, da somit die Sanierungsmaßnahmen des iSFP zielgerichteter erstellt und geplant werden können. Darüber hinaus wurde festgestellt, dass die Formulare nun alle relevanten Felder enthalten, die für die Bearbeitung notwendig sind.
Insgesamt zeigt sich, dass der zweite Prototyp eine solide und stabile Grundlage für die erste produktiv nutzbare Version des Systems außerhalb der Nutzungstests darstellt. Alle definierten Nutzungsanforderungen wurden durch das System erfolgreich erfüllt und nutzerzentriert validiert. Kleinere Verbesserungspotenziale werden in zukünftigen Versionen im Rahmen der weiteren Zusammenarbeit mit dem Praxispartner adressiert.

7.4 Automatisierte Softwaretests

Neben den Benutzungstests, die primär der Validierung des Systems dienen, wurde auch eine softwaretechnische Betrachtung durchgeführt, um die technische Zuverlässigkeit des Systems zu prüfen. Hierzu wurden automatisch ablaufende Softwaretests, genauer gesagt End-to-End-Tests (E2E-Tests) erstellt, die die Funktionsfähigkeit der wichtigsten Anforderungen an die App ganzheitlich verifizieren. Auf die Implementierung dieser wird in Kapitel 6.3.5 eingegangen.
Dazu wird die App auf einem virtuellen Testgerät gestartet und das Nutzungsverhalten eines Nutzers simuliert, indem automatisiert durch die App navigiert und Eingaben getätigt werden, um abschließend zu prüfen, ob das System sich im erwarteten Endzustand befindet. Diese automatisierten Tests tragen wesentlich dazu bei, die Qualität und Zuverlässigkeit der App sicherzustellen.

8

Fazit und Ausblick

8.1 Fazit

Die Zielsetzung dieser Arbeit war die Entwicklung eines offline-fähigen Softwaresystems zur Erstellung von Vor-Ort-Gebäudebestandsaufnahmen, das die bestehenden Probleme der handschriftlichen Datenerfassung löst, die Anforderungen der Nutzer erfüllt und in Zukunft beim Praxispartner durch die Mitarbeiter verwendet werden kann.

Im Rahmen der Erforschung verwandter wissenschaftlicher Arbeiten und Softwareprodukte wurde deutlich, dass die digitale Erstellung von Gebäudebestandsaufnahmen wesentliche Vorteile bietet, und eine Eigenentwicklung notwendig ist, um die spezifischen Anforderungen des Praxispartners optimal zu erfüllen.
Mithilfe von Feldstudien wurde der Arbeitsablauf umfassend dokumentiert und analysiert. Mehrere Mitarbeiter wurden bei verschiedenen Projekten begleitet, um den zukünftigen Nutzungskontext der Softwarelösung zu verstehen. Dabei zeigte sich, dass der Arbeitsablauf einige Probleme und Schwächen aufweist, die die Effizienz der Datenerfassung und Projektbearbeitung beeinträchtigen. Dazu gehören die variable Qualität und Vollständigkeit der Bestandsaufnahmen, die zu Mehraufwand in der Projektbearbeitung führen, die zeitaufwändige Nachbereitung der erfassten Daten sowie ein Informationsverlust bei der Arbeitsteilung eines Projekts.
Bei der darauf aufbauenden Nutzungskontextanalyse wurde herausgefunden, dass die Mitarbeiter und zukünftigen Nutzer der Softwarelösung rollenbasiert in zwei Benutzergruppen eingeteilt werden können: die Datenerfasser, deren Nutzungskontext das Vorbereiten und Durchführen von Vor-Ort-Terminen zur Gebäudedatenerfassung hauptsächlich am iPad umfasst, und die Projektbearbeiter, die sich teilweise mit den Datenerfassern überschneiden, jedoch ausschließlich im Büro am Computer arbeiten und den Sanierungsfahrplan mit einem Energieberater-Programm am PC erstellen.
Aus der Analyse wurden allgemeine und nutzergruppenspezifische Nutzungsanforderungen abgeleitet. Allgemein ist eine zentrale Cloud-basierte Datenspeicherung erforderlich, die durch Authentifizierung und Autorisierung den gesicherten Zugriff auf Kunden- und Projektdaten ermöglicht. Für Datenerfasser ist eine

offline-fähige, iPad-basierte, digital strukturierte Datenerfassung mit Apple Pencil-Unterstützung wichtig, einschließlich der Möglichkeit, Fotos zu integrieren und gebäudespezifische Besonderheiten zu dokumentieren. Flexibilität in der Navigation ist dabei essentiell. Für Projektbearbeiter muss die Software auf Desktop-PCs nutzbar sein, die Verantwortungszuordnung der Datenerfassung ermöglichen und die Form der Gebäudedaten mit der Energieberater-Software kompatibel sein.
Auf Basis der Analyse wurde ein Konzept entwickelt und in Form von Wireframes skizziert, das die bestehenden Probleme im Arbeitsprozess adressiert. Neben Seiten und Funktionen zur Kunden- und Projektverwaltung stehen dabei vor allem die Formulare zur technischen Gebäudebestandsaufnahme im Vordergrund. Die Angaben zu einem Gebäude werden in mehrere Formularkategorien zu den einzelnen Bauteilen getrennt und über eine Seitenleiste zugänglich gemacht. Jedes Bauteil verfügt über eine eigene Seite mit einem vordefinierten Formularschema, das auf der bestehenden Vor-Ort-PDF-Checkliste basiert und in Zusammenarbeit mit dem Praxispartner erweitert wurde. Die Formulare enthalten verschiedene Feldtypen, wobei Auswahlfelder eine wichtige Rolle spielen. Bedingte Felder werden verwendet, um die Erfassung relevanter Daten nur bei bestimmten Bedingungen zu aktivieren, was die Übersichtlichkeit erhöht. Zusätzlich ermöglichen duplizierbare Formulargruppen die strukturierte Erfassung der Merkmale beliebig vieler Fenstertypen. Die Handschrifterkennung mit dem Apple Pencil ermöglicht eine schnelle und effiziente Dateneingabe. Die Nachverfolgbarkeit der Dateneingaben wird durch die Anzeige von Bearbeiternamen und Zeitstempeln gewährleistet. Es wurde ebenso die Offlinefähigkeit konzipiert, indem Daten zunächst lokal gespeichert und synchronisiert werden, sobald eine Internetverbindung verfügbar ist. Dies stellt sicher, dass die App auch in Umgebungen mit eingeschränkter Konnektivität zuverlässig genutzt werden kann.
Im Rahmen der Implementierung wurde mit dem UI-Entwicklungskit Flutter eine iOS- und Web-App entwickelt. Diese speichert die erfassten Daten zunächst lokal in einer SQLite-Datenbank und synchronisiert sie bei bestehender Internetverbindung mithilfe des Flutter Packages Brick. Die PostgREST-API erwies sich als notwendige Voraussetzung für die Verwendung von Brick zur Erreichung der Offlinefähigkeit. Es wurde zudem festgestellt, dass Brick bei der Erstellung des Repositories ein angepasster HTTP-Client übergeben werden muss, der sicherstellt, dass die Authentifizierungs-Header zum Zeitpunkt des Sendens der Daten gültig sind, da Anfragen bei zeitverzögerter Synchronisierung sonst fehlschlagen würden. Weiterhin wurden Erweiterungen implementiert, um den Nutzer über den aktuellen Synchronisierungsstatus zu informieren. Konflikte beim Synchronisieren, die zu potenziellem Datenverlust führen könnten, wurden mithilfe von PostgreSQL-Funktionen und Update-Triggern adressiert. Um die umfangreichen Anforderungen an die Formulare umzusetzen, wurde festgestellt, dass die von Flutter zur Verfügung stehenden primitiven Widgets dafür nicht ausgelegt sind und zu unübersichtlichem Code sowie erhöhtem Fehlerpotenzial führen würden. Um dieses Problem zu lösen, wurde ein Flutter Package zur modellbasierten Formularerstellung entwickelt. Dieses Package ermöglicht die Erstellung vieler Formulare, erfüllt die gestellten Anforderungen und hält den Code durch Abstraktionen übersicht-

lich.
Die Evaluation durch Nutzungstests zweier aufeinander aufbauender Prototypen mit verschiedenen Mitarbeitern der beiden Nutzergruppen hat Verbesserungspotenziale und Erkenntnisse aufgezeigt, die in Maßnahmen umgewandelt und umgesetzt wurden. Zusammenfassend wurde gezeigt, dass die App eine umfangreichere und effizientere Datenerfassung vor Ort ermöglicht, was zu einer verringerten Nachbereitungszeit für die Datenerfasser führt. Durch die detaillierte Erfassung der Gebäudedaten vor Ort wird die Notwendigkeit für Projektbearbeiter, Pläne und Fotos hinzuzuziehen oder detaillierte Rückfragen an die Datenerfasser zu stellen, reduziert. Dies beschleunigt die Erstellung des Sanierungsfahrplans und erleichtert die Zusammenarbeit mehrerer Mitarbeiter an einem Projekt. Zudem unterstützen die Formulare zu den Kundenanforderungen die Projektbearbeitung, da die Sanierungsmaßnahmen zielgerichteter erstellt und geplant werden können.
Bei der Datenerfassung vor Ort hat sich vor allem die Verwendung von Auswahlfeldern und der Handschrifterkennung als effizienzsteigernd erwiesen. Die Strukturierung der Gebäudedaten in verschiedene Formulare verbessert die Struktur des Arbeitsablaufs und die Seitenleiste zur Navigation erwies sich als entscheidend für eine flexible und situationsspezifische Abarbeitung durch Datenerfasser und Projektbearbeiter. Zudem ermöglicht das Einbinden eines Feldes für zusätzliche Notizen am Ende jedes Formulars die gebäudespezifische Erfassung von Besonderheiten.
Neben den ausschließlich positiven Aspekten fiel in den Nutzungstests auf, dass die Abarbeitung der Formulare den Fokus auf das Kundengespräch reduziert. Es ist jedoch davon auszugehen, dass sich dies mit zunehmender Nutzung verbessert und die Formulare flüssiger in das Gespräch integriert werden können. Im Vergleich zum bisherigen Ablauf ohne App wurde auch festgestellt, dass die Datenerfassung vor Ort durch die Handschrifterkennung und die zusätzliche textuelle Datenerfassung der Fenstertypen geringfügig länger dauert, dies jedoch durch eine verringerte Nachbereitung- und Bearbeitungszeit kompensiert wird.

Zusammenfassend kann gesagt werden, dass die entwickelte Software die Probleme des bisherigen Arbeitsablaufs erfolgreich löst, wie auch die Evaluation gezeigt hat. Das Abtippen der Daten im Rahmen der Nachbereitung entfällt durch die direkte digitale Erfassung vor Ort. Die Vollständigkeit und Genauigkeit der Datenerfassung wird durch die strukturierten Formulare erhöht, was die Erstellung des Sanierungsfahrplans beschleunigt. Die Zusammenarbeit mehrerer Mitarbeiter an einem Projekt wird dadurch erleichtert, dass die offline erfassten Daten mit einer zentralen Cloud-Datenbank synchronisiert werden und somit zur Weiterbearbeitung im Büro bereitstehen, was ebenfalls im Rahmen der Evaluation validiert wurde.
Diese Arbeit dokumentiert, wie das Softwaresystem konzipiert, entwickelt und im Praxiseinsatz getestet wurde. Die entwickelte App erfüllt die Anforderungen der Nutzer und kann in Zukunft im Arbeitsalltag der Mitarbeiter des Praxispartners eingesetzt werden. Damit sind alle Aspekte der Zielsetzung dieser Arbeit erfolgreich bearbeitet und erreicht.

8.2 Limitationen

Es ist zu beachten, dass alle Ergebnisse und Erkenntnisse dieser Arbeit nur eine Annäherung an die Wirklichkeit darstellen. Die Nutzungstests bieten eine wertvolle erste Einschätzung, jedoch ist die Aussagekraft eingeschränkt, da sie unter der besonderen Situation eines Nutzungstests einer neuen Software durchgeführt wurden, was zu einer gründlicheren Arbeitsweise der Tester führen könnte. Zudem ist die gesamte Nutzergruppe mit vier Personen relativ klein, was zu subjektiven Erkenntnissen führen kann. Aufgrund der Auftragslage und des begrenzten Zeitrahmens konnte besonders die Projektbearbeitung nur eingeschränkt getestet werden.
Ebenso sind die Projekte und Gebäude sehr unterschiedlich, weshalb die App in folgenden Anwendungen ihre Zuverlässigkeit und Gebrauchstauglichkeit weiterhin unter Beweis stellen muss. Aussagen über die langfristige Gebrauchstauglichkeit der App in den verschiedenen Nutzungskontexten können erst nach einer längeren Nutzung im routinierten Arbeitsalltag getroffen werden.

8.3 Ausblick

Die vorliegende Arbeit hat eine solide Grundlage für ein im Praxisalltag einsatzbereites, offline-fähiges Softwaresystem zur technischen Gebäudebestandsaufnahme geschaffen. Die gewonnenen Ergebnisse sind vielversprechend, und die Software hat in den Nutzungstests gezeigt, dass die Probleme des bisherigen Arbeitsablaufs durch die Software gelöst werden. Es gibt jedoch zahlreiche Möglichkeiten zur Weiterentwicklung und Optimierung der Software.
Wie in der Arbeit beschrieben, konnte die Fotodokumentation aus zeitlichen Gründen nicht implementiert werden. Hierbei sind offene Fragen zu den genauen Implementierungsdetails zu klären, wie beispielsweise die lokale Speicherung der Fotos und deren anschließende Synchronisation mit den weiteren Daten. Bisher wurden nur Daten in Form von JSON synchronisiert, jedoch keine Fotos als Dateien.
Darüber hinaus gab es in den letzten Nutzungstests noch kleinere Verbesserungsmöglichkeiten, die weiter bearbeitet werden können, um die App zu optimieren. Die Konfliktbehandlung, die während der Entwicklung zwar berücksichtigt, aber nicht vollständig erforscht wurde, bietet ebenfalls Potenzial für weitere Verbesserungen. Beispielsweise könnte untersucht werden, wie weitere Arten von Konflikten bei der Synchronisierung erkannt und behandelt werden können.
Neben diesen technischen Erweiterungen ist es essenziell, den Einsatz der Software langfristig zu evaluieren. Dies wird im Rahmen der weiteren Zusammenarbeit mit dem Praxispartner geschehen, um sicherzustellen, dass die Software langfristig einen großen Mehrwert für die Optimierung der Arbeitsprozesse bietet und diese somit effizienter gestaltet.

Literaturverzeichnis

BAF. *Bundesförderung Energieberatung für Wohngebäude.* https://www.bafa.de/DE/Energie/Energieberatung/Energieberatung_Wohngebaeude/energieberatung_wohngebaeude_node.html [abgerufen am 15.06.2024].

BKKB21. Borrmann, André, Markus König, Christian Koch und Jakob Beetz (Herausgeber): *Building information modeling: Technologische Grundlagen und industrielle Praxis.* VDI-Buch. Springer Vieweg, Wiesbaden and Heidelberg, 2., aktualisierte Auflage Auflage, 2021.

Cha. Chapps AG. https://www.chapps.com/de/products/objektbegehungen-app/ [abgerufen am 08.07.2024].

Cou. Courier Plus, Inc.: *An intuitive way to work with persistent data in Dart.* https://github.com/GetDutchie/brick [abgerufen am 03.07.2024].

CRS. Cudok, Falk, Felix Rehmann und Rita Streblow: *BIM im Gebäudebestand – Herausforderungen in der Sanierung.*

Fas. FastField Inc.: *FastField Forms | Mobile Data Collection and Analytics.* https://docs.github.com/en/actions/learn-github-actions [abgerufen am 03.07.2024].

Git. GitHub, Inc.: *Learn GitHub Actions - GitHub Docs.* https://docs.github.com/en/actions/learn-github-actions [abgerufen am 03.07.2024].

Gooa. Google LLC: *Build apps for any ccreen.* https://flutter.dev [abgerufen am 03.07.2024].

Goob. Google LLC: *Firebase Crashlytics.* https://firebase.google.com/docs/crashlytics [abgerufen am 03.07.2024].

ISO18. *Ergonomie der Mensch-System-Interaktion – Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte (ISO 9241-11:2018)*, November 2018.

ISO20. *Ergonomie der Mensch-System-Interaktion – Teil 210: Menschzentrierte Gestaltung interaktiver Systeme (ISO 9241-210:2019)*, März 2020.

KWvM19. Kleppmann, Martin, Adam Wiggins, Peter van Hardenberg und Mark McGranaghan: *Local-first software: you own your data, in spite of the cloud.* In: Masuhara, Hidehiko und Tomas Petricek (Herausgeber): *Proceedings of the 2019 ACM SIGPLAN Interna-*

tional Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Seiten 154–178, New York, NY, USA, 2019. ACM.

Pos. PostgreSQL Global Development Group: *PostgreSQL: The World's Most Advanced Open Source Relational Database.* `https://www.postgresql.org/` [abgerufen am 03.07.2024].

PSS+22. Pereira, Clara, João N. Silva, Ana Silva, Jorge de Brito und José D. Silvestre: *Building Inspection System Software Based on Expert Knowledge.* Journal of Performance of Constructed Facilities, 36(2), 2022.

SQL. SQLite Consortium: *SQLite Home Page.* `https://www.sqlite.org/` [abgerufen am 03.07.2024].

Sup. Supabase Inc: *Supabase | The Open Source Firebase Alternative.* `https://supabase.com` [abgerufen am 03.07.2024].

Abkürzungsverzeichnis

API Application Programming Interfaces. 42, 47, 51

BIM Building Information Modeling. 3, 4
BLoC Business Logic Component. 46–48

CRM Customer-Relationship-Management. 22, 28

GUI Grafical User Interface. 38, 47, 60

HTTP Hypertext Transfer Protocol. 51

iSFP individueller Sanierungsfahrplan. 14–16, 19, 20, 22, 23, 57–61, 69, 70

JSON JavaScript Object Notation. 42, 50
JWT JSON Web Token. 42, 67

REST Representational State Transfer. 42, 50, 51
REST-API Representational State Transfer Application Programming Interface. 39, 42, 45, 46, 50–52
RLS Row Level Security. 43

SDK Software Development Kit. 42, 46, 50, 51
SQL Structured Query Language. 37, 39, 41, 43

UI User Interface. 37, 45, 46, 48, 51