Projektdokumentation: PhileTipTip App und Phileteirus Admin Tool

Matthias Hochmuth

3. November 2024

Inhaltsverzeichnis

1 Einleitung

Über das Beispielunternehmen Philetairus Immobilien GmbH

Abbildung 1: Logo der Philetairus Immobilien GmbH,

Firmenname	Philetairus Immobilien GmbH
Gesellschaftsform	GmbH
Branche	Immobilienindustrie
Gründungsjahr	1979
Standorte Limburg an der Lahn, Mainz	Wiesbaden(Hauptsitz),
Produzent oder Dienstleister	Dienstleister
Anzahl Mitarbeiter	550

Tabelle 1: Steckbrief Philetairus Immobilien GmbH

Hauptdienstleistungen Die Philetairus Immobilien GmbH ist ein Komplettdienstleister im Bereich der Immobilienwirtschaft. Das Kerngeschäft umfasst die Verwaltung von Wohnungseigentum (WEG-Verwaltung) sowie die Miet- und Sondereigentumsverwaltung.

Weitere Dienstleistungen beinhalten technisches, kaufmännisches und infrastrukturelles Gebäudemanagement, Vermietungsmanagement, Erstellung der Betriebskostenabrechnungen sowie Hausmeisterservices.

Firmenziele Die Philetairus Immobilien GmbH strebt danach, einen umfassenden Service für die zu verwaltenden Immobilien zu bieten, der sowohl Eigentümer als auch Mieter zufriedenstellt. Ein weiteres Ziel ist die Förderung einer starken Kooperation der einzelnen

Unternehmensbereiche, um einen zentralen Anlaufpunkt für alle Bedürfnisse der Auftraggeber zu bieten.

Die hier beschriebene Firma ist rein fiktiv und wurde nur für den Zweck der Habmann Abschlussarbeit erstellt und wurde im Rahmen dieses Moduls unverändert übernommen.

1.1 Projektbeschreibung

Kurze Einführung in das Projekt, Zielsetzung und Nutzen.

1.1.1 Einführung

Das Projekt PhileTipTip zielt darauf ab, ein innovatives System für die Abteilung Außenarbeiten zu entwickeln. Dieses System soll in Form einer Android-App realisiert werden und dient als Kommunikationsbrücke zwischen Mietern/Eigentümern und der Grünflächenabteilung einer Immobilienverwaltung. Zusätzlich soll ein Administrations Tool für die Desktop Anwendung entstehen, die das erfassen von neuen Datensätzen für Mieter, Gebäude und Mitarbeiterdaten vereinfacht und vereinheitlicht. Für dieses Pionierprojekt sollen beide Komponenten der Erfassung von Meldungen für die Grünflächenabteilung dienen, aber perspektivisch soll mit diesem Projekt der Grundstein für eine umfassende Digitalisierung des gesamten Unternehmens gelegt werden.

1.1.2 Zielsetzung

Das Hauptziel des Projekts ist es, einen effizienten und benutzerfreundlichen Weg zu schaffen, um Meldungen über Probleme im Außenbereich von Immobilien zu erfassen, zu verwalten und zu bearbeiten. Insbesondere soll das System folgende Funktionen bieten:

- Ermöglichung der Erfassung von Meldungen durch Mieter und Eigentümer
- Fokus auf Probleme im Zuständigkeitsbereich der Abteilung Außenarbeiten (z.B. Schädlingsbefall, Unkrautwuchs)
- Effiziente Weiterleitung der Meldungen an die zuständige Abteilung
- Verwaltung und Bearbeitung der eingegangenen Meldungen durch die Abteilung Außenarbeiten

1.1.3 Nutzen

Die Implementierung dieses Systems bringt mehrere Vorteile mit sich:

- Verbesserte Kommunikation zwischen Bewohnern und Immobilienverwaltung
- Schnellere Reaktionszeiten auf Probleme im Außenbereich
- Erhöhte Effizienz in der Bearbeitung von Meldungen
- Bessere Übersicht und Nachverfolgbarkeit von Außenarbeiten
- Steigerung der Zufriedenheit von Mietern und Eigentümern

1.1.4 Kernfunktionen

Die Android-App PhileTipTip wird folgende Kernfunktionen beinhalten:

- Benutzerfreundliche Oberfläche zur Erfassung von Meldungen
- Möglichkeit zum Hochladen von Fotos zur besseren Dokumentation der Probleme
- Kategorisierung der Meldungen (z.B. Schädlingsbefall, Unkrautwuchs)
- Automatische Weiterleitung der Meldungen an die zuständige Abteilung
- Statusverfolgung der Meldungen für Benutzer
- Verwaltungsinterface für die Bearbeitung und Aktualisierung von Meldungen

1.2 Zielgruppe

Die primären Nutzer der App sind:

- Mieter von Immobilien
- Eigentümer von Immobilien
- Mitarbeiter der Abteilung Außenarbeiten in der Immobilienverwaltung

1.3 Ziele des Projekts

Durch die Entwicklung von PhileTipTip wird ein modernes, effizientes System geschaffen, das die Kommunikation und Problemlösung im Bereich der Außenarbeiten von Immobilien signifikant verbessert.

2 Projektorganisation

2.1 Auswahl der Projektstruktur

Bevor mit der Planung des angestrebten Projekts begonnen werden kann, muss zunächst eine grundlegende Entscheidung über das Vorgehen und das Projektmanagementmethode getroffen werden. Da die verschiedenen Methoden bereits bei der Projektplanung teilweise bedeutende Unterschiede aufweisen, muss diese Entscheidung bereits vor der Planungsphase getroffen werden.

2.2 Kurzer Vergleich gängiger Projektmanagementoptionen

Grundsätzlich wird beim Projektmanagement zwischen drei Ansätzen unterschieden:

Klassisch	Linear Sequentiell. Erst wenn die Arbeiten eines Schrittes abgeschlossen wurden, wird mit dem nächsten begonnen. Die Abfolge und Abläufe der Schritte sind statisch und ein Abweichen vom Projektplan ist nicht ohne weiteres Möglich. Bei Projekten dieser Art wird viel Zeit auf eine sehr detailierte Vorplanung gelegt, die dann über fest definierte Meilensteine überprüft und gesteuert werden kann. Ein bekannter Vertreter dieses Ansatzes ist das Wasserfallmodell.
Agil	Inkrementell Iterativ. Während des Projektes wird das Endprodukt immer weiter verfeinert. Die Vorplanung bei Projekten dieser Art ist im Vergleich zum klassischen Vorgehen deutlich weniger detailiert. Viel mehr werden die Anforderungen in ständiger Kommunikation mit dem Kunden/Auftraggeber kontinuierlich überprüft und verfeinert, in der Regel nach dem Erreichen von Zwischenschritten (Inkrementen). Ein bekannter Vertreter dieses Ansatzes ist Scrum.
Hybrid	Der Hybride Ansatz verbindet Elemente des Klassischen und des Agilen Projektmanagements um sich so die Vorteile beider Vorgehensweisen zu Nutze zu machen. Etwa indem die umfassende Projektstruktur klassisch aufgebaut ist um eine solide Struktur und klare Planbarkeit sicherzustellen, aber auf der Teamebene die Dynamik des agilen Ansatzes nutzt um bei der Erstellung des eigentlichen Projektprodukts flexibel agieren zu können. Eine Möglichkeit diesen Ansatz umzusetzen ist Prince2 (diese Methode bietet auch die Option rein klassisch vorzugehen)

Tabelle 2: Kurzübersicht Projektmanagementoptionen

2.3 Agiles Vorgehen im Projekt

Eine etablierte Möglichkeit festzustellen, welches Vorgehen sich für ein konkretes Projekt eignet ist die Stacey Matrix.

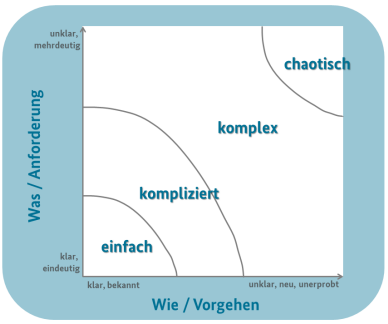

Abbildung 2: Stacey Matrix (https://www.bva.bund.de/DE/Services/Behoerden/Beratung/Beratungszentrum/GrossPM/Wissenspool/_documents/Standardartikel/stda-stacey-matrix.html)

Im Bezug auf das geplante Projekt sind die Anforderungen noch nicht wirklich klar ausformuliert. Die Geschäftsführung hat eine Vision, eine grobe Idee und keinen klaren Katalog an sorgfältig definierten Anforderungen. Die Anforderungen sind also unklar.

Das Vorgehen an sich ist innerhalb des organisatorischen Ökosystems der Philetairus Immobilien GmbH unklar und unerprobt. Die IT Abteilung hat zuvor noch keine größeren Entwicklungsprojekte durchgeführt, daher ist auf diesem Gebiet wenig Erfahrung vorhanden und die Einbindung und Einarbeitung der neuen Mitarbeiter kommt noch hinzu. Es gibt keine Erfahrungswerte innerhalb des Unternehmens auf die zurückgegriffen werden könnte, das Vorgehen ist daher unklar und unerprobt.

Basierend auf diesen Parametern ist dieses Projekt als Komplex einzuschätzen, daher ist ein agiles Vorgehen ratsam um der bestehenden Unklarheit möglichst flexibel zu begegnen und während des Projektes die geeigneten Methoden und Techniken zu erarbeiten. Für ein klassisches Vorgehen wäre eine langwierige Vorbereitungsphase notwendig, in der die genauen Anforderungen ermittelt werden müssten, bevor das Projekt überhaupt starten könnte.

Ein weiterer Vorteil des agilen Vorgehens in diesem Fall besteht in der zuvor beschriebenen Iterativen und Inkrementellen Vorgehensweise. So kann in regelmäßigen Abständen der Fortschritt überprüft werden und gegebenenfalls Anpassungen und Ergänzungen an den Anforderungen vorgenommen werden. Prinzipiell wäre das auch bei einem traditionellen Ansatz möglich, allerdings ist es dort mit einem größeren Aufwand verbunden einmal fixierte Projektanforderungen zu ändern .

Diese Flexibilität steht in direktem Zusammenhang mit der engen Einbindung der (in diesem Fall unternehmensinternen) Auftraggeber, die typisch für agiles Projektmanagement ist. Während die Auftraggeber bei klassischen Vorgehen konkrete Anforderungen zu

Beginn des Projektes definieren und diese dann entweder zum Ende (oder zu festgelegten Meilensteinen) prüfen und abnehmen, stehen der Auftraggeber (oder dessen Vertreter) bei agilen Vorgehen in ständiger Kommunikation mit dem Entwicklerteam, prüfen die Zwischenprodukte (Inkremente) und beteiligen sich aktiv an der Anpassung und Aktualisierung der Anforderungen (zum Beispiel über das Product Backlog, siehe 3.3).

Zu der Flexibilität des Vorgehens und des geringen Aufwands der Projektvorbereitung kommt ein weiterer Vorteil hinzu - die vergleichsweise hohe Anzahl der neuen Mitarbeiter, die durch die Erweiterung der IT Abteilung hinzugekommen sind. Diese sind noch nicht vollständig in die Unternehmensstrukturen integriert, auch hier bietet der agile Ansatz den Vorteil, da er auf flache Hierachien und Zusammenarbeit auf Augenhöhe setzt und die neuen Mitarbeiter somit schneller in das Ökosystem integriert werden können.

Aus diesen Gründen wurde der Entschluss gefasst, das Projekt agil anzugehen. Allerdings gab es von Seiten der Geschäftsleitungen einige Bedenken bezüglich eines rein agilen Vorgehens, weshalb weitere Überlegungen zur Projektstruktur angestellt wurden.

2.4 Hybrides Vorgehen im Projekt

Während die Stacey Matrix hilft das Projekt zu beurteilen und eine Entscheidung für klassisches oder agiles Vorgehen im Projekt zu treffen, gibt es andere Methoden und Hilfsmittel, den Projektkontext, nicht das Projekt als solches zu betrachten und einzuordnen. Eines davon ist das Agilometer:

Abbildung 3: Agilometer (https://www.pureconsultant.de/de/agile/agilometer/)

Betrachtet man die Philetairus Immobilien GmbH, so ist der agile Reifegrad äußerst

niedrig. Es gibt das Bestreben, zumindest auf der für das vorliegende Projekt relevanten Ebene agiler zu agieren, doch die mangelnde praktische Erfahrung bei der umsetzung von agilen Projekten, die immer noch bestehende Skepsis der Unternehmensführung und die große Anzahl von neuen Mitarbeitern in der ausführenden Abteilung deuten darauf hin, dass ein rein agiles Vorgehen keine sinnvolle Wahl wäre.

Ein klassisches Vorgehen nach dem Wasserfall- oder V-Modell wäre aufgrund der noch recht unkonkreten Zielsetzung des Projektes auch wenig praktikabel, da für diese Vorgehensweise mit Lasten- und Pflichtenheft die Anforderungen möglichst konkret und unmissverständlich formuliert werden müssen, bevor das Projekt initiiert werden kann.

Aus diesem Grund ist für das Vorhaben ein Hybrides Vorgehen, welches klassisches und agiles Vorgehen miteinander verbindet ideal. Für das agile Vorgehen fiel die Entscheidung auf Scrum (siehe 2.3), den klassischen Rahmen soll Prince2 bilden.

Prince2 gibt klare Vorgaben, wie das Projektmanagementteam strukturiert sein sollte, für die Planung auf der Teamebene gibt es allerdings keine starren Richtlinien. Daher ist es möglich, ein hybrides Vorgehen zu wählen. Im Fall dieses Projekts wurde die Entscheidung für diese diese Variante getroffen: traditionelles Vorgehen auf der Entscheidungsebene und agiles Vorgehen auf der operativen Ebene. Somit bleibt ein großes Maß an Kontrolle erhalten, ohne die Vorteile des agilen Vorgehens zu sehr zu beschneiden.

Auf der anderen Seite definiert der Scrum Guide sehr genau die Rollen und Verantwortlichkeiten auf der Teamebene sowie das Vorgehen während der Entwicklung, die genaue Einbindung in die Unternehmensorganisation wird allerdings nicht konkret vorgegeben, somit ergänzen sich die beiden Vorgehensweisen, da es keine Überlappungen gibt, die Änderungen an einem der beiden Vorgehensmodellen nach sich ziehen und Kompromisse erfordern würden.

Ein weiterer Vorteil des Hybriden Vorgehens ist, dass gerade bei Prince2 durch die hohe Anzahl an Projektmanagementprodukten der Dokumentation von Erfahrungswerten ein hoher Wert beigemessen wird, die bei einem rein agilen Vorgehen eine untergeordnete Rolle spielt, wie es im bereits im agilen Mainifest niedergeschrieben wurde:

Funktionierende Software mehr als umfassende Dokumentation

Zwar bedeutet das nicht, dass nicht dokumentiert werden sollte, allerdings wird dem einzelnen Inkrementen die in den Sprints erstellt werden ein höherer Stellenwert beigemessen. Das hybride Vorgehen ermöglicht es, diesem Gedanken gerecht zu werden, indem ein

Großteil der Dokumentationsarbeit auf die Projektmanagementebene ausgelagert wird, wodurch das Entwicklerteam entlastet wird, aber dennoch eine umfassende Dokumentation der Erfahrungswerte gewährleistet wird.

3 Anforderungsanalyse

3.1 Anforderungsanalyse

Auch wenn die Entscheidung getroffen wurde, das Projekt hybrid und in diesem Rahmen die eigentliche Entwicklung agil umzusetzen ist es dennoch ratsam, eine gewisse Vorarbeit zu treffen um zum einen den Business Case für den Prince2 Rahmen als auch die User Stories für das Product Backlog (siehe 3.3) entwickeln zu können. Im Kontext der Softwareentwicklung wird dieser Prozess als Anforderungsanalyse (Requirements Engineering) bezeichnet. In diesem Prozess werden die Anforderungen erfasst, analysiert, strukturiert und priorisiert. Außerdem werden in dieser Phase erste Spezifikationen festgelegt, die als Rahmen für den weiteren Projektverlauf dienen können.

Ebenfalls wichtig ist es, gewisse Abgrenzungskriterien zu erfassen um bereits vor Beginn der Arbeiten sicher festzuhalten, was nicht Ziel des Projekts ist. Auf diese Weise wird zum einen ein sogenannter Scope- oder Featurecreep vermieden, der dafür sorgen würde, dass immer weitere, aber nicht benötigte oder geforderte Funktionalitäten eingebaut werden. Zum anderen wird auf diese Weise sichergestellt, dass plötzliche Anforderungen von Auftraggeberseite aus den ursprünglich definierten Projektrahmen nicht zu stark erweitern.

Durch das agile Vorgehen, den iterativ inkrementellen Lieferansatz und den stetig aktualisierten Product Backlog ist hierbei eine detailierte Ausarbeitung nicht so entscheidend, wie bei einem sequentiell durchgeführten Projekt, dennoch ist es ratsam auch hier sorgfältig vorzugehen um späteren Missverständnissen und Verzögerungen vorzubeugen. In einem rein nach Prince2 durchgeführten Projekt ohne eine agile Komponente auf der Lieferebene würden diese Anforderungen in direkt die Baseline einfließen, was zur Folge hätte, dass Abweichungen von dieser Baseline erst nach Genehmigung eines Änderungsantrags durchgeführt werden könnten. Durch das agile Vorgehen wird hier etwas Flexibilität gewonnen, da die Anforderungen in das Product Backlog überführt werden, dass weniger starr aufgebaut ist wie ein Lastenheft, aber dennoch einen konkreten Rahmen benötigt.

Das Ziel des ersten Schritts der Digitalisierung der Philetairus Immobilien GmbH wurde nur eine grobe Projektidee umrissen. Gemäß dieser Idee soll ein System für die

Abteilung Aussenarbeiten geschaffen werden, welches es Mietern und Eigentümern ermöglicht Meldungen abzusenden, die in das Aufgabengebiet dieser Abteilung fallen (z.B. Schädlingsbefall oder Unkrautwuchs). Neben dem Senden der Meldungen wurde die Verwaltung und Bearbeitung dieser Meldungen als weitere Anforderung genannt. Als Arbeitstitel für das Projekt wurde PhileTipTip gewählt, in Anlehnung an den Warnruf des Siedelwebervogel, der dem Unternehmen Philetairus Immobilien GmbH seinen Namen verliehen hat.

Da die geplante Software einen bereits bestehenden Geschäftsprozess unterstützen und optimieren soll, ist es wichtig diesen Prozess so gut wie möglich zu verstehen. Eine Möglichkeit dafür ist das Modellieren eines Geschäftsprozesses, um die Abläufe, Akteure und Zusammenhänge zu erfassen. Diese Modellierung kann dann auch als Grundlage für die Erstellung von UML Diagrammen für die konkrete Entwicklung verwendet werden.

Der Erfassen und Verstehen dieses Prozesses findet in enger Zusammenarbeit mit den Stakeholdern statt, die später die Anwender der zu entwickelnden Software sind. Der Einsatz des im Projekt erstellten Programms soll den bestehenden Prozess so gut es geht optimieren, Redundanzen reduzieren, Reaktionszeiten erhöhen und Fehler vermeiden und auf diese Weise zum einen Kosten senken und zum anderen die Zufriedenheit der Beteiligten am Prozess erhöhen.

3.2 Geschäftsprozessmodellierung

Eine Möglichkeit der Visualisierung ist die Geschäftsprozessmodellierung oder Business Process Modelling, die in natürlicher Sprache die Abläufe des zu beschreibenden Prozesses aufzeigt. Gerade im Bereich der Digitalisierung von Abläufen und Vorgängen ist diese Betrachtung sinn- und wertvoll, da sie zum einen einen konkreten Einblick in die eigentlichen Prozesse schafft und somit eventuelle Optimierungsansätze erkennen lässt und zum anderen für die Entwickler der Digitalisierungslösung einen ersten Überblick gestattet, aus dem Use-Cases und User Stories extrahiert werden können.

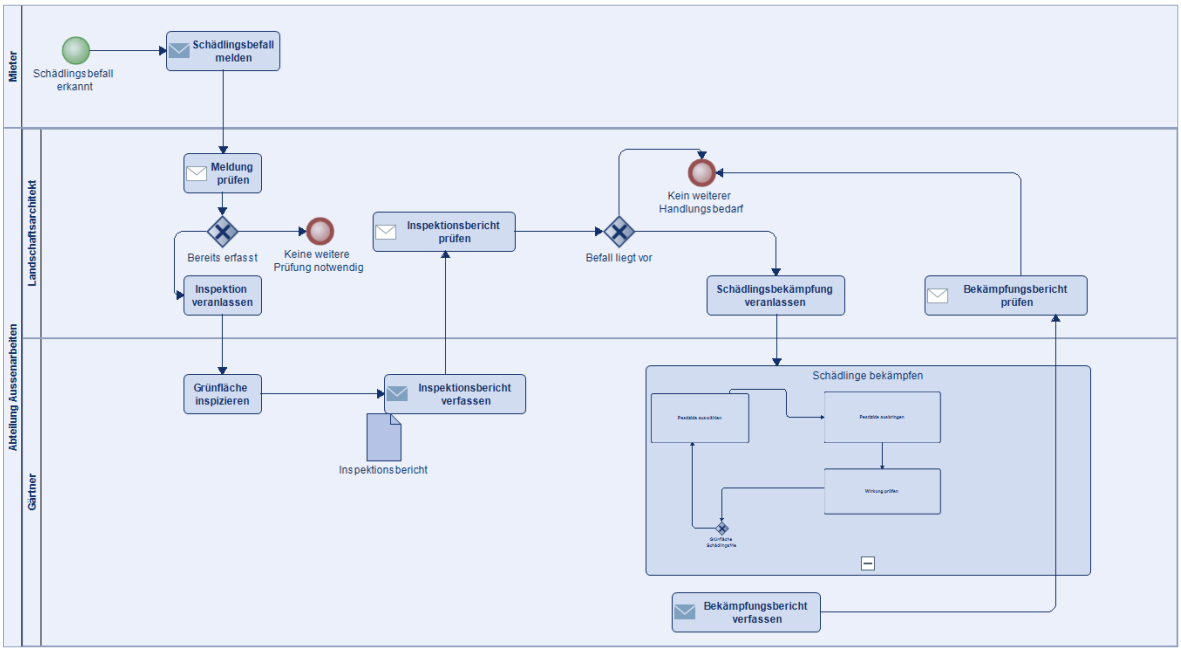

Abbildung 4: BPM Diagramm PhileTipTip - eigene Darstellung, erstellt mit Modelio Version 5.4.01

Das (vereinfachte) BPM Diagramm stellt den bisherigen Ablauf dar, von der Meldung eines Schädlingsbefalls oder Unkrautbewuchses. Anhand dieser Grafik ist erkennbar, dass es in diesem Prozess viele Optimierungsmöglichkeiten gibt, die sowohl die benötigte Zeit, als auch die an den jeweiligen Schritten beteiligten Akteure reduzieren können. Diese Optimierung entlasten die jeweilige Abteilung, vermeiden Redundanzen und sparen somit Geld und Zeit.

Nachdem der Prozess an sich erfasst und verstanden wurde, können die Anforderungen konkretisiert werden. Es ist ersichtlich, dass es drei Hauptakteure gibt, die für die weitere Planung berücksichtigt werden müssen, unter anderem bei der Erstellung von User Stories und Use Cases. Diese User Stories (siehe 3.3.1) werden für die weitere Be-

arbeitung im Product Backlog (siehe 3.3) erfasst und bilden die Basis für den ersten/die ersten Sprints des Scrum Teams.

Sowohl die User Stories als auch das Use Case Diagram können und werden sich im Laufe des Projekts (mehrfach) ändern, da es durch das inkrementelle und iterative Vorgehen sehr wahrscheinlich ist, dass einige spezielle Anforderungen und Nutzerwünsche erst ab einem gewissen Entwicklungsschritt ersichtlich werden. Durch die kontinuierliche Kommunikation mit den Endbenutzern ist es möglich, diese während der Entwicklungsphase zu erfassen und umzusetzen.

3.2.1 Funktionale und Nichtfunktionale Anforderungen

Für gewöhnlich wird in einer Anforderungsanalyse zwischen Funktionalen und Nichtfunktionalen Anforderungen unterschieden. Funktionale Anforderungen beziehen sich auf den Funktionsumfang des Programms, was soll es tun. Nicht-Funktionale Anforderungen können vereinfacht ausgedrückt als Qualitätsanforderungen bezeichnet werden (wie schnell ist der Zugriff, wie sicher die Verschlüsselung, wie stabil die Verbindung etc.).

Um die Anforderungen zu erfassen, müssen diese Anforderungen erst gesammelt und erfasst werden (hauptsächlich durch Gespräche mit den unterschiedlichen Stakeholdern). Nach Abschluss dieses Schrittes müssen diese Anforderungen analysiert werden, um sie in eine Hierachie und einen Bezug zueinander zu bringen und zu ermitteln, ob sich einzelne Anforderungen unterstützen (Zielkomplementarität), behindern (Zielkonkurrenz) oder sogar ausschließen (Zielantinomie).

Die Funktionalen Anforderungen beantworten die Frage Was das Programm leisten soll.

- Erfassung von Meldungen über Schädlingsbefall
- Speichern der Meldungen
- Anzeige und Sortierung dieser Meldungen
- Weiterleitung der Meldungen an die Grünflächenabteilung
- Benachrichtigungen an Nutzer über Bearbeitungsstatus

Die Nichtfunktionalen Anforderungen beantworten die Frage: Wie soll das Programm diese Anforderungen erfüllen?

- Das Business verlangt, dass die Daten On-Premises, also auf Firmeninterner Hardware gehalten werden [**?**], nicht aufgrund belastbarer Fakten oder Erfahrungswerte, sondern aufgrund eines starken Misstrauens gegenüber der Cloud Technologie und der Angst vor Kontrollverlust. Von dieser Haltung ist der Projektauftraggeber nicht abzubringen.
- Einfache Bedienbarkeit der App
- Leichte Wartbarkeit der App
- Leichte Erweiterbarkeit der App
- Zuverlässiges Funktionieren der App
- Datensicherheit und Datenschutz
- Schnelle Reaktionszeiten
- Niedriger Datenverbrauch im Mobilbetrieb

3.2.2 Verständliche Pläne - Domain Driven Design und Geschäftsprozessmodellierung

Bei der Erstellung von Individualsoftware ist ein anderes Vorgehen erforderlich als etwa für Standardsoftware, die einen breiten Markt ansprechen und abdecken soll. Durch die Verwendung von Prince2 in Kombination mit Scrum auf der Teamebene ist eine starke Kommunikation zwischen den Auftraggebern/Endanwendern beziehungsweise deren Stellvertretern und den Entwicklern, sowie ein effetkiver Austausch der Entwickler untereinander, ausschlaggebend für den Erfolg.

Damit diese Kommunikation effektiv und für alle Stakeholder transparent erfolgen kann, bietet es sich an, hier auf bewährte Diagramme zurückzugreifen, die sämtlichen interessierten Stakeholdern zugänglich gemacht werden können, etwa über einen Information Radiator. Auch wenn nicht beabsichtigt wird, einen vollkommen Modellgetriebenen Ansatz für die Entwicklung der App zu wählen, wird dennoch bereits in der Vorbereitung der Entschluss gefasst, stark auf Modelle und Diagramme zu setzen um die technischen Zusammenhänge für alle Stakeholder schlüssig darzustellen und die logischen Zusammenhänge und Interaktionen der einzelnen Prozesse und Akteure für die Entwickler verständlich zu visualisieren und so eine solide Kommunikationsbasis zu schaffen.

Die Möglichkeit noch einen Schritt weiterzugehen als die Modellgetriebene Softwareentwicklung wäre das sogenannte *Domain Driven Design*, auf das in diesem Fall allerdings verzichtet wird, da die Domäne (noch) nicht komplex genug ist um den Aufwand der Modellierung zu rechtfertigen. Allerdings werden (auch durch den agilen Ansatz) viele Kernpunkte dieser Vorgehensweise in die Entwicklung von PhileTipTip einfließen.

3.3 Product Goal und Product Backlog

Der Scrum Guide ordnet jedem Artefakt ein Commitment (eine Verpflichtung) zu. Im Falle des Product Backlog ist es das Product Goal. Das Produkt-Ziel oder Product Goal ist das gesetzte Ziel, auf dessen Erreichen sich das Scrum Team verpflichtet (commited) hat. Das Product Goal ist das langfristige Ziel für das Scrum Team. Das Scrum Team muss ein Ziel erfüllen (oder aufgeben), bevor es das nächste angeht. Das Product Goal beschreibt einen zukünftigen Zustand des Produkts, welches dem Scrum Team als Planungsziel dienen kann. Das Product Goal befindet sich im Product Backlog. Der Rest des Product Backlogs entsteht, um zu definieren, was das Product Goal erfüllt [**?**].

Vereinfacht kann man das Product Goal wörtlich als Produktziel übersetzen. Es definiert, in welche Richtung sich das Produkt Sprint für Sprint entwickeln soll. Gemeinsam mit den User Stories wird es im Product Backlog festgehalten, aus dem einzelne Einträge für die Sprints im Sprint Planning ins Sprint Backlog überführt und dann bearbeitet werden.

Da nur Einträge, die in einem Sprint erledigt (Done) werden können zur Auswahl für das Sprint Backlog geeignet sind, müssen diese vorbereitet werden, etwa indem zu große User Stories oder Epics in kleinere, präzisere Elemente zerlegt werden. Diese Aktivität wird als Backlog Refinement bezeichnet und ist ein kontinuierlicher Vorgang, da das Product Backlog kein statischer Zustand, sondern dynamisch ist um dem agilen Ansatz gerecht zu werden.

Es ist die Aufgabe des Product Owners (siehe **??**) das Produkt-Ziel zu entwickeln und zu kommunizieren, die Product Backlog Einträge zu erstellen, durch das festlegen einer Reihenfolge zu priorisieren und diese klar zu kommunzieren.

Neben der Planung der Arbeiten innerhalb eines Sprints (siehe **??**) dient das Product-Goal auch als Maß um den Fortschritt des Projekts (etwa im Sprint Review - siehe **??**) zu prüfen, denn jedes aus einem Sprint resultierende Increment soll dazu dienen, sich dem Produktziel anzunähern. Das Product Goal fördert somit die Transparenz und den Fokus hinsichtlich des Fortschritts des Product Backlogs, so wie das Sprint-Ziel das für das Sprint Backlog tut.

In diesem Fall deckt sich das initiale Product Goal mit der Grundidee, die für die App PhileTipTip formuliert wurde. Für die Verwendung als Product Goal wurde diese Idee ein wenig straffer formuliert und einzelne Details, die sich in der ursprünglichen Beschreibung noch fanden, ausgelassen. Diese werden in Form von User Stories und Einträgen im

Product Backlog festgehalten.

Product Goal PhileTipTip:
Schaffung eines digitales Systems, das es Mietern und Eigentümern ermöglicht, Auffälligkeiten auf Grünflächen einfach und effizient zu melden, um eine gezielte Instandhaltung durch das Außenteam sicherstellen zu können.

Der Scrum Guide gibt nicht explizit vor, wie die Einträge auszusehen haben, mit denen das Product Backlog gefüllt werden soll, aber ein bewährtes Vorgehen ist die Verwendung von User Stories, die in kurzen, gut verständlichen Sätzen darlegen, Wer Was Warum möchte. Aus diesen User Stories (die, wenn sie größer ausfallen als Epics bezeichnet und vor der Bearbeitung zerlegt werden müssen) setzt sich dann (zusammen mit dem Product Goal) das Product Backlog zusammen.

3.3.1 User Stories

Als [Rolle]	möchte ich [Funktionalität]	damit [Grund]
Mieter / Eigentümer	Schädlingsbefall / Unkrautbewuchs melden können	die Gärtner diesen zeitnah beseitigen
Landschaftsarchitekt	Mietermeldungen einsehen können	zu analysieren und Handlungsanweisungen geben zu können
Landschaftsarchitekt	Meldungen als Arbeitsanweisung an mein Team weiterleiten können	arbeiten zu koordinieren
Landschaftsarchitekt	einsehen können, woran mein Team gerade arbeitet	zu priorisieren und gegebenenfalls einzugreifen
Gärtner	einsehen können, wo ich arbeiten soll	den Einsatzort schnell finde
Gärtner	einsehen können, woran ich arbeiten soll	die Arbeit vorbereiten kann (passenede Herbi/Pestizide)
Abteilungsleiter Aussenarbeiten	einsehen können, woran die verschiedenen Teams gerade arbeiten	Arbeiten zu koordinieren und Absprachen mit den Landschaftsarchitekten halten zu können.
Abteilungsleiter Aussenarbeiten	die Meldungen über Befall und Bewuchs einsehen können	eventuelle Muster zu erkennen und großflächige Ausbreitung von Unkraut und Schädlingen zu verhindern.

Tabelle 3: Ausgewählte User Stories

Die erste User Story kombiniert zwei Anwendungsfälle und zwei Rollen, damit sich

die Entwickler bewusst sind, dass sie diese Funktionalität sowohl für Mieter als auch für Eigentümer zur Verfügung stellen müssen. Noch steht nicht konkret fest, wie sich die Anforderungen dieser beiden Nutzergruppen unterscheiden, das kann sich durch Backlog Refinement oder durch in Sprint Reviews gewonnenen Erfahrungen noch ändern. In diesem Fall müsste diese Story konkretisiert oder aufgespalten werden.

Ebenso verhält es sich mit dem zweiten Teil - die Funktionalität Schädlingsbefall oder Unkrautbewuchs melden zu können wurde in einer einzelnen User Story erfasst. Auch hier könnte es notwendig sein, diese Funktionalität zu spalten doch im Bestreben das Product Backlog übersichtlich zu halten, wurden diese Angaben zunächst in einer gemeinsamen Story erfasst.

Die nachfolgenden Stories stehen jeweils für sich, sie beziehen sich jeweils auf einzelne Rollen und Anwendungsfälle, aber auch in diesen Fällen kann ein Refinement im Laufe des Projekts notwendig sein.

4 App-Design

4.1 UI/UX

- Benutzeroberfläche (UI) für einfache Erfassung.
- Übersichtliche Navigation und Menüstruktur.
- Bild-Upload-Funktion zur Dokumentation.
- Klare, intuitive Navigation mit großen, leicht erkennbaren Buttons.
- Einsatz der Corporate-Farben: Dunkles Grün (23423d) für Hauptelemente und Wheat (f5deb5) für Hintergründe und Akzente.
- Integration des Webervogel-Logos als zentrales Designelement, z.B. als App-Icon und in der Kopfzeile.
- Bottom-Navigation-Bar mit maximal 4-5 Hauptkategorien für schnellen Zugriff.
- Einfacher Kamera-Button für direktes Fotografieren oder Auswahl aus der Galerie.
- Vorschau-Funktion mit Möglichkeit zur Bildbearbeitung (Zuschneiden, Drehen).
- Einfache Bedienbarkeit
- Step-by-Step Meldevorgang mit Fortschrittsanzeige.

- Autocomplete-Funktion für Adresseingabe und Problembeschreibung.
- Haptisches Feedback bei wichtigen Aktionen für bessere Nutzerinteraktion.

4.2 Design-Richtlinien

- Konsistente Verwendung der Corporate-Farben 23423d und f5deb5 in allen UI-Elementen.
- Einheitliche Schriftart und -größen für optimale Lesbarkeit.
- Verwendung von Icons im Stil des Webervogel-Logos für eine harmonische visuelle Identität.
- Responsive Design für verschiedene Bildschirmgrößen und -orientierungen
- Verwendung der offiziellen Google Icons für ein einheitliches Bild und bewährtes Design https://fonts.google.com/icons

5 Technische Architektur

6 Überlegungen zum Technologiestack und der Toolchain

Für die Umsetzung der geplanten Anwendungen gibt es mehre Optionen. Die eingesetzten Technologien, Programmiersprachen und Frameworks bezeichnet man als Technologiestack oder Techstack. Da die Entwicklungsabteilung der IT neu aufgebaut wird, handelt es sich hierbei auch um eine Grundsatzentscheidung, denn obwohl es möglich ist, Folgeprojekte mit einem anderen Techstack durchzuführen hat ein etabliertes und bewährtes Konzept viele Vorteile, vor allem die gewonnen Erfahrung der Mitarbeiter.

6.1 Auswahl der Programmiersprache

Da viele der Entscheidungen bezüglich der einzusetzenden Anwendungen und Technologien von der verwendeten Programmiersprache Abhängig sind, muss diese Entscheidung getroffen werden, bevor weitere Überlegungen angestellt werden können. Durch den geplanten Anwendungsfall und die Ausbildung der Mitarbeiter der Informationstechnologie Abteilung wurde die Auswahl auf Objektorientierte Programmiersprachen beschränkt, so werden Sprachen, die einem anderen Paradigma folgen wie zum Beispiel funktionale Sprachen nicht in Betracht gezogen.

Die Programmiersprache für den Kern der Anwendungen muss einige Anforderungen erfüllen:

- Plattformunabhängigkeit (PC, Android und IOS Systeme)
- Stabilität und Sicherheit
- Zukunftssicherheit
- Flexibilität und leichte Zugänglichkeit

Aufgrund dieser Anforderungen wurden die neuen Mitarbeiter der Informationstechnologie Abteilung zu einem Workshop eingeladen, der im Vorfeld des eigentlichen Projekts stattfand. Aufgrund der Tragweite der zu treffenden Entscheidung wurde das Format des Lightning Talk gewählt, bei dem jeder Mitarbeiter seine bevorzugte Programmiersprache vorstellen und deren Vorteile herausstellen konnte, moderiert wurde dieser Workshop von Frau Mareike Strauss, die darauf achtete, dass die festgelegte Redezeit nicht überzogen wurde und jeder Teilnehmer gleichberechtigt seine Meinung einbringen konnte.

Unter den vorgestellten Sprachen befanden sich unter anderem:

- Python, C++, Kotlin
- Javascript, Java, C#

Python hat insbesondere durch die rasante Entwicklung der KI in den letzten Jahren an Bedeutung gewonnen, die Sprache ist relativ leicht zu erlernen und durch eine große Menge an verfügbaren Plugins und Erweiterungen auf viele Anwendungsfälle anpassbar. Allerdings erfüllt Python nicht die gewünschte Zukunftssicherheit, da bereits beim Wechsel von Version 2.7 auf 3.0 Änderungen vorgenommen wurden, die die Abwärtskompatibilität opfern um Optimierungen an der Sprache vorzunehmen. Für den geplanten langfristigen Einsatz einer komplexen Codebasis ist eine solide Kompatibilität über Versionen hinweg jedoch wichtig, da sonst ein notwendiges Update auf eine neue Version langwierige und fehleranfällige Anpassungen nach sich ziehen würde.

C++ ist eine gut entwickelte und mächtige Programmiersprache, die vor allem für Hardwarenahe Programmierung eingesetzt wird. Allerdings wird sie (gerade im Vergleich zu moderneren Programmiersprachen als umständlich und kompliziert empfunden. Viele der Teilnehmer schreckte gerade das Konzept der Zeiger und der Header ab, was im Vergleich zu anderen vorgestellten Sprachen von den Workshopteilnehmern als umständlich und kompliziert empfunden wurde. Zudem kann die Speicherverwaltung in C++ gerade für unerfahrene Programmierer schnell zu Fehlern führen, die die gewünschte Stabilität

der App gefährden würden.

Ein weiterer Vorschlag war es, PhileTipTip als Web-App auf Javascript Basis unter Verwendung von unterschiedlichen Javascript Frameworks (wie React und Next.js) für Front- und Backend umzusetzen. Ein Vorteile bei diesem Vorgehen ist, dass die App unabhängig von Apple und Android auf Mobilgeräte veröffentlich werden kann, da die Nutzer über einen Browser ihrer Wahl auf eine Webseite zugreifen, ohne eine App aus dem Playstore (Android) oder App Store (Apple) herunterladen zu müssen. Gegen diesen Vorschlag spricht der höhere Aufwand, sowohl die Scripte an sich als auch den Server gegen Zugriffe und Manipulation abzusichern, was zwar möglich, aber mangels Erfahrung der Mitarbeiter nicht ohne weiteres umsetzbar wäre.

Die Entscheidung fiel nach einer Abstimmung auf Java. Zum einen ist die weite Verbreitung und die Stabilität ein großer Vorteil, es gibt zahlreiche ausgereifte Entwicklungsumgebungen und Plugins für viele Problemstellungen. Durch die große Community ist es sehr wahrscheinlich, schnelle Hilfestellung auf eventuelle Probleme und Antworten auf auftretende Fragen zu erhalten.

Ein weiterer Vorteil ist, dass Java gerade im Vergleich zu C++ relativ einsteigerfreundliche, aber dennoch mächtige Programmiersprache ist. Gegenüber Python ist die stärkere Objektorientierung des Sprachkerns ein Vorteil für die Planung und Umsetzung der Systeme, außerdem bietet der LTS (Long Term Support) von Oracle eine Zukunftssicherheit, die bei Versionswechseln von Python nicht unbedingt vorrausgesetzt ist. Gegenüber Kotlin liegen die Vorteile an der weiten Verbreitung und langen Historie der Sprache, die große Community und zahlreichen Frameworks und Plugins. Sollte in der Zukunft die Entscheidung getroffen werden, auf Kotlin umzusteigen wäre dieser Schritt durch die Interoperationalität möglich.

Obwohl C# Java in vielen Aspekten stark ähnelt wurde die Entscheidung dennoch für Java getroffen, da nicht geplant wurde, das .Net Ökosystem und verwandte Microsoft Technologien (wie Asp.Net oder Azure) einzusetzen. Durch die starke Ähnlichkeit der Sprachen ist ein Wechsel für zukünftige Projekte, die eine dieser Technologien verwenden sollen mit geringem Aufwand möglich.

6.2 Die Toolchain

Neben der Programmiersprache und der Datenbanktechnologie gilt es, vor dem Beginn der eigentlichen Entwicklungsarbeit noch einige Entscheidungen bezüglich der einzusetzenden

Programme, der Toolchain zu treffen.

6.2.1 Die IDE

Die IDE - Integrated Developement Environment (Integrierte Entwicklungsumgebung), diese enthält neben einem Codeeditor mit farblicher Markierung von Schlüsselworten zahlreiche für die Programmierer hilfreiche Funktionalitäten, die sowohl die Entwicklung als auch die Fehlersuche (das Debugging) erleichtern und beschleunigen. So können zum Beispiel Haltepunkte (die sogenannten Breakpoints) gesetzt werden um an bestimmten Stellen die Ausführung des Programms kontrolliert zu unterbrechen und den Programmfluss Schritt für Schritt nachvollziehen zu können.

Prinzipiell wäre es natürlich möglich, dass jeder Entwickler eine individuelle Entscheidung bezüglich der Entwicklungsumgebung trifft, aber auch vor dem Hintergrund des Pair Programming ist es sinnvoll, auf eine einheitliche Lösung zurückzugreifen. In diesem Fall soll für die Entwicklung der App Android Studio eingesetzt werden , eine bewährte Java IDE, die mit einer Vielzahl von Plugins angepasst und erweitert werden kann und bereits für die Entwicklung von Android Apps optimiert ist.

Für die Entwicklung des Desktop basierten Administrationstools eignet sich Android Studio nur bedingt, daher wird an dieser Stelle auf IntelliJ IDEA zurückgegriffen, welches wie Android Studio von Jetbrains entwickelt wird und somit keine erneute Einarbeitung der Entwickler in eine grundlegend andere IDE (wie zum Beispiel Eclipse) erfordert. Nahezu sämtliche Tastaturkürzel die die Arbeit in Android Studio erleichtern und beschleunigen stehen auch in IntelliJ zur Verfügung und auch die Aufteilung des Arbeitsbereichs ist vertraut und verringert die Transferzeiten und Einarbeitung der Mitarbeiter.

6.2.2 Versionierung

Ein Versionskontrollsystem erlaubt es mehreren Entwicklern zeitgleich an der selben Codebasis zu entwickeln und Änderungen an den selben Dateien vorzunehmen. Hierfür wird ein zentraler Server verwendet, auf dem der Main-Branch des Projekts liegt, der die Basis für alle verbundenen Entwicklerrechner dient. Hat ein Programmierer eine Teilfunktion abgeschlossen, kann er diese Änderung mit der auf dem Server zusammenführen, wobei Änderungen klar hervorgehoben werden und im Problemfall wieder rückgängig gemacht werden können. Versionskontrollsysteme bieten neben dieser Basisfunktionalität noch weitere hilfreiche Funktionen, wie etwa das Anlegen von mehreren Branches, automatisiertes Builden des Codes und weiteres. Für dieses Projekt wird Git [**?**] eingesetzt, eine weit verbreitete Lösung für Versionskontrolle. Um nicht auf die Kommandozeile zurückgreifen

zu müssen wird GitGui als grafisches Tool verwendet.

Im Gegensatz zu der IDE ist es hier allerdings den Entwicklern freigestellt, ja nach Vorliebe ein alternatives Tool wie zum Beispiel GithubDesktop oder Sourcetree zu verwenden, oder auf Plugins in der IDE zurückzugreifen, da diese Tools lediglich eine einfach bedienbare Oberfläche als Alternative der Kommandozeilenbefehle für die Versionierung zur Verfügung zu stellen. Diese Tools bieten alle einen einfachen Überblick über Änderungen und können auch zur Fehlersuche genutzt werden, indem der fehlerhafte Code mit früheren Versionen verglichen wird.

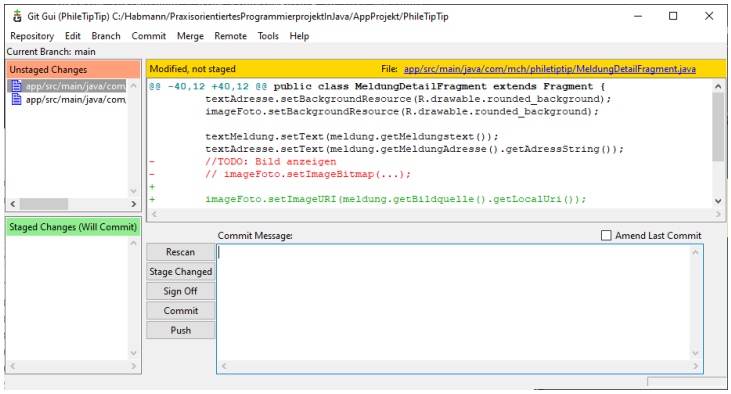

Abbildung 5: Git Gui - Grafisches Tool zur Versionsverwaltung

Ein weiterer Vorteil, der sich aus der Verwendung eines Versionierungssystems eröffnet ist die Möglichkeit, einen CI/CD Prozess einzubinden. CI steht in diesem Fall für Continous Integration und bedeutet grob gesagt, dass bei jeder Codeänderung die ein Entwickler in das Versionierungssystem hochläd automatische Prozesse losgetreten werden - in der Regel handelt es sich um einen Build Prozess, der diese Änderungen zusammen mit den aktuellen Änderungen der anderen beteiligten Entwickler zusammen zu einer ausführbaren Anwendung zusammensetzt und in der Regel auch vordefinierte Texts durchführen wird, um zu verhindern, dass fehlerhafte und nicht funktionale Codeänderungen in das Endprodukt einfließen.

Das CD kann, je nach gewählten Ansatz entweder für Continous Delivery oder Continous Deployment stehen. In beiden Fällen entsteht nach dem CI Schritt ein ausführbares Produkt, was sowohl im Sinne der Scrum Philosophie als auch des Prince2 Prozesses ist. Der Hauptunterschied besteht darin, dass bei Continous Delivery dieses Zwischenprodukt (welches ein Scrum Inkrement darstellen könnte) zunächst manuell freigegeben werden muss, während es bei Continous Deployment sofort nach bestehen der vordefinierten Tests an die angebundenen Nutzer ausgeliefert wird, wodurch diese ständig Zugriff auf die aktuellste Version haben, wodurch die Feedbackschleife deutlich kürzer gestaltet

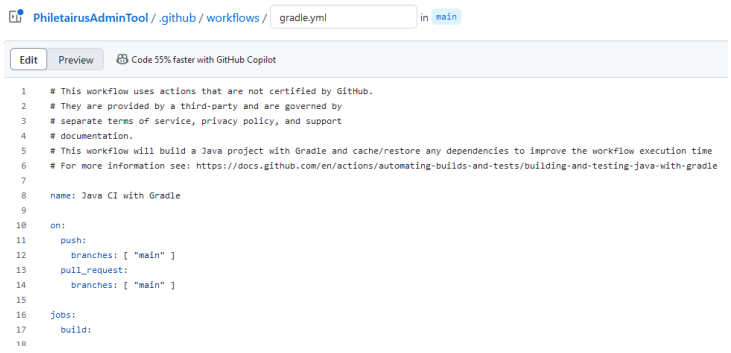

Abbildung 6: Github Actions

wird, aber unter Umständen auch Zwischenstände von Funktionen, die noch nicht für den Anwendungstest vorgesehen sind veröffentlicht werden können, in beiden Fällen ist eine durchdachte Kommunikationsstrategie notwendig.

6.2.3 Kommunikation und Projektmanagement

Ein weiteres Tool auf das sich die Projektbeteiligten geeinigt haben ist Jira [**?**] - ein Tool über das Tasks verwaltet werden können und das sich in der agilen Softwareentwicklung bewährt hat. Mit Jira können Tickets zu einzelnen Userstories angelegt, bearbeitet und zugeteilt werden, im Sinne der Selbstorganisation natürlich auch von den Entwicklern selbst. Dieses System erleichtert den Überblick für den Product Owner, schafft Übersicht für die Entwickler und auch für den Lenkungsausschuss sowie Projektmanager. Ein weiterer Nebeneffekt ist, dass durch diese digitale Lösung auf Kanban-Boards oder ähnliches verzichtet werden kann wodurch der Nachhaltigkeitsansatz von Prince2 unterstützt wird.

6.2.4 Frameworks und Bibliotheken

Frameworks sind Sammlungen von Funktionalitäten, die als Paket in das jeweilige Projekt eingebunden werden, um häufig wiederkehrende Probleme und Herausforderungen zu lösen. Häufig vereinfachen Sie Zugriffe auf andere Bibliotheken, indem sie komplizierte Aufrufe vor dem Entwickler verbergen und eine vereinfachte Implementierung gestatten. Eines dieser Frameworks, welches zum Einsatz kommt ist Jitpack, welches eng mit der Versionsverwaltung über Git zusammenarbeitet.

Jitpack erlaubt es, die in einem Repository (Sammlung der hochgeladenen Dateien) befindlichen Klassen und Funktionen in mehreren Projekten einzubinden, ein entscheidender Vorteil für die zweigeteilte Entwicklung PhileTipTip und Phileteirus Admin Tool, die beide auf gemeinsame Datenklassen (etwa für Mieter, Meldungen und Projekte) zugreifen. So müssen Änderungen nur an einer Stelle vorgenommen werden und es kann

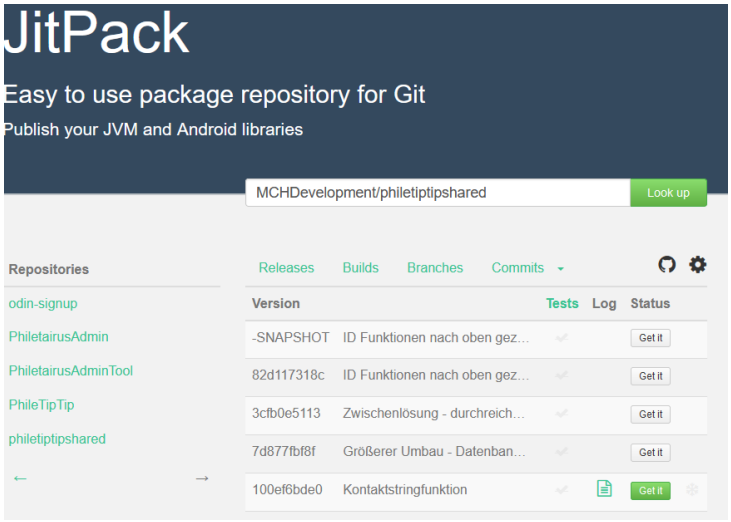

Abbildung 7: Jitpack

nicht zu Inkomptibilitäten oder Redundanzen kommen.

Neben diesem Framework werden weitere bewährte Lösungen in der Entwicklungen eingesetzt, wie zum Beispiel JavaFX für die Entwicklung der grafischen Oberfläche des Admintools, welches durch die Gestaltung der GUI mithilfe von FXML in Verbindung mit Controller Klassen ein sehr ähnliches Vorgehen wie Android Studio mit der gestaltung über XML Dateien in Verbindung mit Activities ode Fragmenten zur Kontrolle verfolgt.

FormsFX und ValidatorFX werden für das Erstellen und Verifizieren von Formularen eingesetzt und das bewährte Spring Framework hilft, eine Struktur bereitzustellen, die modernen Ansprüchen gerecht wird und spätere Anbindungen an weitere Dienste und Erweiterungen erleichtert.

6.3 Systemarchitektur

6.3.1 Client-Server-Modell

Sowohl die App PhileTipTip als auch das dazugehörige Philetairus Admin Tool basieren auf einem klassischen Client-Server-Modell, bei dem die App als Client fungiert und über eine API mit einem zentralen Backend kommuniziert. Der Client (die App beziehungsweise die Desktopanwendung) sind für die Datenerfassung durch die Nutzer verantwortlich, während das Backend die Verarbeitung, Speicherung und Verwaltung dieser Daten übernimmt. Dieses Modell ermöglicht eine klare Trennung der Verantwortlichkeiten zwischen der Benutzeroberfläche und der Datenverarbeitung, was zu einer besseren Skalierbarkeit und Flexibilität führt.

6.3.2 Integration der Datenbank

Die MySQL-Datenbank ist fest in das Backend integriert und dient als zentrale Speicherinstanz für die erfassten Meldungen, sowie der Datensätze für Mieter, Mitarbeiter, Wohnanlagen und Immobilien. Jede Anfrage des Clients, die eine Datenveränderung oder -abfrage erfordert, wird vom Backend an die Datenbank weitergeleitet. Über definierte API-Endpunkte können Nutzer der App Meldungen zu Schädlingsbefall oder Unkrautbewuchs erstellen und den Status dieser Meldungen abfragen. Gleichzeitig erlaubt das Backend der Grünflächenabteilung, auf diese Daten zuzugreifen, sie zu bearbeiten und den Bearbeitungsstatus zu aktualisieren.

Die enge Verzahnung beider Anwendungen ermöglicht es leicht, auf dieser Datenbankstruktur weitere Anwendungen aufzusetzen, um das Vorhaben der vollständigen Digitalisierung des Unternehmens Schritt für Schritt umzusetzen, wobei der Modulare Ansatz und der Modellcharakter dieses Projekts eine solide Basis bilden und die gewonnenen Erfahrungen in weitere Entwicklungen einfließen können.

6.3.3 Backend und API

Das Backend wird als Vermittler zwischen der App und der Datenbank fungieren. Es nimmt die Anfragen des Clients entgegen, verarbeitet sie und stellt die entsprechenden Daten bereit. Die API, die auf REST-Prinzipien basiert, stellt sicher, dass die Kommunikation zwischen der App und dem Server effizient und sicher erfolgt. Zu den wichtigsten Aufgaben des Backends gehören:

- Verarbeitung von Nutzeranfragen: z.B. das Erstellen neuer Meldungen oder das Abrufen bestehender Einträge.
- Sicherheit und Authentifizierung: Schutz der Daten durch Zugriffskontrollen und verschlüsselte Kommunikation.
- Datenmanagement: Verwaltung und Speicherung der Daten in der MySQL-Datenbank sowie Sicherstellung der Datenintegrität.

Diese Architektur sorgt für eine flexible und robuste App, die auf wachsende Nutzerzahlen und Anforderungen skalierbar ist. Zudem ermöglicht sie eine klare Trennung zwischen Frontend (App) und Backend (Datenverarbeitung), was die Wartung und Weiterentwicklung der App erleichtert.

6.4 Technologien

6.4.1 MySQL-Datenbank

Für die Verwaltung der Anwendungsdaten wird eine MySQL-Datenbank eingesetzt. MySQL ist ein weit verbreitetes relationales Datenbankmanagementsystem, das sich durch seine Zuverlässigkeit, hohe Performance und Skalierbarkeit auszeichnet. Die Datenbank speichert alle wichtigen Informationen, wie Benutzerdaten, Meldungen von Schädlingsbefall oder Unkrautbewuchs und deren Bearbeitungsstatus. Die Anbindung erfolgt über eine API, die die Kommunikation zwischen der App und der Datenbank ermöglicht.

6.5 Schnittstellen

6.5.1 API zur Kommunikation zwischen Frontend und Backend

Die Kommunikation zwischen dem Frontend (der Android-App) und dem Backend erfolgt über eine RESTful API. Diese API ermöglicht eine klare und strukturierte Interaktion zwischen den beiden Komponenten, indem sie Endpunkte bereitstellt, über die die App auf die vom Backend verwalteten Daten zugreifen kann. Jede Aktion, die von der App ausgeführt wird – sei es das Erfassen einer neuen Meldung von Schädlingsbefall oder das Abrufen des Status einer bestehenden Meldung – erfolgt über HTTP-Anfragen an die API.

Die wichtigsten API-Methoden umfassen:

- POST: Zum Erstellen neuer Meldungen, die von Nutzern erfasst werden.
- GET: Zum Abrufen von Daten, wie z.B. dem Bearbeitungsstatus einer Meldung.
- PUT: Zum Aktualisieren von Daten, etwa wenn die Grünflächenabteilung den Status einer Meldung ändert.
- DELETE: Für das Löschen von nicht mehr relevanten Daten.

Die API ist dabei so konzipiert, dass sie sowohl eine hohe Performance als auch eine sichere Kommunikation gewährleistet. Dies erfolgt durch die Implementierung von HTTPS zur Verschlüsselung der Datenübertragung und einer Token-basierten Authentifizierung, die den Zugriff nur für berechtigte Nutzer und Systeme ermöglicht.

6.5.2 Benachrichtigungssysteme

Neben der reinen Datenkommunikation bietet die App ein Benachrichtigungssystem, das die Nutzer über den Status ihrer Meldungen informiert. Diese Benachrichtigungen werden durch das Backend ausgelöst, wenn bestimmte Ereignisse eintreten, wie z.B.:

- Eingangsbestätigung einer Meldung: Sobald ein Nutzer eine Meldung abschickt, erhält er eine Bestätigung, dass die Daten erfolgreich erfasst wurden.
- Status-Updates: Sobald die Grünflächenabteilung eine Meldung bearbeitet oder den Status ändert, wird der Nutzer per Push-Benachrichtigung informiert.
- Erinnerungen: Falls eine Meldung über einen längeren Zeitraum unbeantwortet bleibt, können Erinnerungen an das Bearbeitungsteam oder die Nutzer gesendet werden.

Diese Benachrichtigungen werden über Firebase Cloud Messaging (FCM) versendet, das eine zuverlässige und effiziente Zustellung von Push-Benachrichtigungen an die Android-Geräte der Nutzer sicherstellt. So bleiben die Nutzer jederzeit über den Stand ihrer Meldungen informiert, ohne aktiv in der App nachsehen zu müssen.

7 Implementierung

7.1 Codierstandards

7.2 Best Practices und Namenskonventionen

Die Entwicklung der App PhileTipTip folgt klar definierten Codierstandards, um die Lesbarkeit, Wartbarkeit und Skalierbarkeit des Codes sicherzustellen. Die Einhaltung dieser Richtlinien ermöglicht eine konsistente Struktur des Quellcodes, erleichtert die Zusammenarbeit im Team und minimiert potenzielle Fehlerquellen.

7.2.1 Konsequentes Einsetzen der Objektorientierung und Entwurfsmuster

Die App wird vollständig nach dem Prinzip der Objektorientierten Programmierung (OOP) entwickelt. Dies bedeutet, dass alle funktionalen Bereiche in Klassen und Objekte unterteilt werden, um die Wiederverwendbarkeit und Modularität zu gewährleisten. Zudem kommen bewährte Entwurfsmuster wie das Singleton-Pattern (für die zentrale Verwaltung der Datenbankinstanz) und das Factory-Pattern (zur dynamischen Objekterstellung) zum Einsatz, um typische Aufgabenstellungen effizient zu lösen und den Code robust und flexibel zu halten.

7.2.2 Modularisierung und Anwendung der SOLID-Prinzipien

Ein zentrales Element der Architektur ist die strikte Modularisierung des Codes. Jede Funktionalität wird in klar abgegrenzten Modulen implementiert, die für sich unabhängig getestet und weiterentwickelt werden können. Diese Trennung fördert die Wiederverwendbarkeit von Code und erleichtert Erweiterungen. Zusätzlich wird die Entwicklung konsequent an den SOLID-Prinzipien ausgerichtet:

- Single Responsibility Principle (SRP):Jede Klasse erfüllt nur eine klar definierte Aufgabe.
- Open-Closed Principle (OCP):*Klassen und Module sind offen für Erweiterungen, aber geschlossen für Änderungen, um unnötige Anpassungen im bestehenden Code zu vermeiden.
- Liskov Substitution Principle (LSP): Objekte von Unterklassen können durch Objekte der Oberklasse ersetzt werden, ohne dass das Verhalten der Anwendung beeinträchtigt wird.
- Interface Segregation Principle (ISP): Schnittstellen werden klein und spezifisch gehalten, um unnötige Abhängigkeiten zu vermeiden.
- Dependency Inversion Principle (DIP): Abhängigkeiten werden auf Abstraktionen statt auf konkrete Implementierungen aufgebaut, um die Flexibilität des Codes zu erhöhen.

7.2.3 Kommentar- und Formatierungsrichtlinien

Um den Code für alle Entwickler verständlich und nachvollziehbar zu gestalten, werden klare Kommentar- und Formatierungsrichtlinien beachtet. Kommentare erklären nicht nur den Zweck des Codes, sondern auch komplexe Abläufe, Algorithmen oder wichtige Entscheidungen bei der Implementierung. Dabei wird insbesondere auf die Prägnanz und Relevanz der Kommentare geachtet.

Bei der Formatierung folgen wir gängigen Konventionen, wie:

- Einheitliche Einrückungen (z.B. 4 Leerzeichen pro Ebene).
- Sinnvolle Benennung von Variablen und Methoden nach dem CamelCase-Format.
- Konsistenter Einsatz von Leerzeilen und Absätzen zur logischen Gliederung des Codes.
- Begrenzung der Zeilenlänge, um die Lesbarkeit auf verschiedenen Bildschirmen zu gewährleisten.

Durch diese Maßnahmen wird sichergestellt, dass der Code nicht nur funktional korrekt ist, sondern auch für andere Entwickler leicht zu verstehen und weiterzuentwickeln ist.

7.3 Feature-Entwicklung

Schwerpunkte und iterative Entwicklung.

7.4 Tests

- Unit-Tests und UI-Tests.
- Teststrategie für die App.

8 Qualitätssicherung

8.1 Testplan

Zeitplan für Tests und Fehlerbehebung.

8.2 Testumgebung

Beschreibung der Geräte und Android-Versionen.

8.3 Abnahmekriterien

Anforderungen für den Produktivstart.

9 Datenschutz und Sicherheit

9.1 Sicherheit und Datenschutz

Datensicherheit und Datenschutz sind gerade dann wenn, wie im Falle von PhileTip-Tip, personenbezogene Daten erhoben, verarbeitet und gespeichert werden von äußerster Wichtigkeit. Die Risiken, die mit diesen Themen verbunden sind, sollten daher noch einmal gesondert betrachtet werden und den jeweiligen Risikoeigentümern und Risikobearbeitern bewusst gemacht werden.

Betrachtet man die App gesondert von der Datenbank und dem Admin Tool kommt noch eine weitere Besonderheit hinzu - die Zugriffsrechte auf Systemfunktionen und Speicherstruktur des Smartphones des jeweiligen Nutzers. Da der Zugriff auf die GPS Koordinaten, die die Bestimmung des genauen Standorts erlauben und der Kamera sowie dem internen Speicher Rückschluss und Zugriff auf sensible Daten gestatten, ist es wichtig, diese Eingriffe so gering und für den Nutzer so nachvollziehbar wie möglich zu halten.

Die Abfrage an der Stelle, an der die Funktionalität benötigt wird und nicht gleich zum Aufstarten der App schafft Vertrauen beim Benutzer, da transparent wird, welche Funktion des Geräts für welchen Schritt in der Erfassung benötigt wird. Nach dem Grundsatz der Datensparsamkeit werden auch wirklich nur benötigte Funktionen abgefragt -

Abbildung 8: Zugriffsrechtabfrage - eigene Darstellung, Screenshot der PhileTipTip App

das Popup, welches GPS Zugriff anfordert wird ein Nutzer nicht zu Gesicht bekommen, wenn er die Adresse selbst einträgt, ebenso wird ein Nutzer, der ein Bild aus seiner Gallerie läd nach Zugriff auf den Medienspeicher seines Smartphones gefragt, allerdings nicht nach dem Zugriff auf die Kamera.

Softwareseitige Risiken, die vom Entwicklerteam berücksichtigt und vermieden werden müssen sind unter anderem: SQL Injections, die ein Risiko für die Integrität der Datenbank darstellen. Bei dieser Art des Angriffes wird über ein unzureichend abgesichertes Formularelement eine ausführbare MySql Anweisung übertragen, die dann auf dem Server interpretiert wird und dort entweder Zugangsüberprüfungen umgeht, Daten ausließt, manipuliert oder beschädigt. Diese Art des Angriffs ist weit verbreitet, aber auch gut zu verhindern, sofern sich die Entwickler dieser Gefahr bewusst sind [**?**].

Hardwareseitige Risiken, die von der IT-Abteilung berücksichtigt und vermieden werden müssen sind unter anderem: DDOS Angriffe: Bei dieser Art des Angriffs wird der attackierte Server mit einer Vielzahl an Anfragen überlastet, bis seine Kapazitäten erreicht sind und er unerreichbar wird.
Serverausfälle: Auch ohne externe Angriffe kann es zu Serverausfällen kommen, etwa

durch Hardwareschäden oder Stromausfälle. Redundante Kopien (im Idealfall räumlich getrennt) und regelmäßige Backups begrenzen die Auswirkungen dieser Schäden und eine USV (Unabhängige Stromversorgung) stellt sicher, dass zumindest kurzfristige Stromausfälle überbrückt werden können.
Ein Offlinemodus der App, bei dem die Daten auf den lokalen Geräten vorgehalten werden, bis der Server wieder online ist ist ebenfalls eine Möglichkeit Datenverlust bei den Nutzern zu vermeiden, da diese die Erfassung der Schadstelle wie gewohnt durchführen können und das hochladen auf den Server verschoben wird, bis dieser wieder betriebsbereit ist.

Menschliche Risiken, die von den Anwendern berücksichtigt und vermieden werden müssen sind unter anderem: Social Engineering ist eine Art des Angriffs, der weder auf Hard- noch auf Software, sondern auf den Mensch im System abzielt. Hierbei gibt sich der Angreifer zum Beispiel als Mitglied des IT-Supports aus und verlangt die Herausgabe des Passworts um so unberechtigt Zugriff auf die Daten zu erlangen. Um gegen diese Art des Angriffs geschützt zu sein ist es ratsam, ein gewisses Bewusstsein für Sicherheit bei den firmeninternen Nutzern zu schaffen, zum Beispiel durch Schulungen und klare Abläufe, wie der Support Kontakt zu den Mitarbeitern aufnimmt.

Bei den Nutzern außerhalb des Firmenökosystems (den Mietern und Eigentümern) sollte zumindest ein Hinweis eingeblendet werden, dass Phileteirus Immobilien GmbH niemals nach dem Passwort fragen wird um diese Gefahr zumindest abzumildern. Auch hier ist der Ansatz der Datensparsamkeit von hoher Wichtigkeit - da viele relevante Daten über Mieter, Mitarbeiter und Immobilieneigentümer bereits aus anderen Quellen vorliegen, können diese Nutzer sensibilisiert werden. Wenn sie es gewohnt sind, dass die App keinerlei Daten abfragt und anfordert, die nicht direkt mit dem Erfassungsprozess verbunden sind, werden sie im Falle eines Angriffs misstrauischer und vorsichtiger agieren und eventuelle Angriffe und Datenlecks schneller erkennen und melden.

Juristische Risiken, die von der Projektleitung und der Rechtsabteilung berücksichtigt und vermieden werden müssen sind unter anderem: Persönlichkeitsrechte - Recht
Die PhileTipTip App erlaubt es den Nutzern Fotos von Schäden an Grünflächen anzufertigen um die Bewertung durch die Abteilung Außenabteilungen zu erleichtern. Daher ist es möglich, dass Personen auf den Bildern zu erkennen sind, die dem nicht zugestimmt haben. Hier muss auf eine juristisch haltbare Lösung hingearbeitet werden, etwa durch klar formulierte Allgemeine Nutzungsbedingungen der App, denen der Nutzer verbindlich zustimmt und somit selbst die Verantwortung übertragen bekommt die Privatsphäre Dritter zu wahren und zu achten.

Generell wird die Digitalisierung des Unternehmens, je weiter sie voranschreitet immer komplexer werden und die Berufung eines Datenschutzbeauftragten wird ab einem gewissen Punkt nicht nur sinnvoll, sondern notwendig werden um DSGVO (Datenschutzgrundverordnung) Konform zu handeln und somit rechtssicher auftreten zu können.

10 Risiken und Herausforderungen

10.1 Risikomanagement

Identifikation potenzieller Risiken und Maßnahmen zur Risikominimierung.

10.2 Notfallpläne

Vorgehen bei schwerwiegenden Fehlern oder Ausfällen.

11 Abschluss und Ausblick

11.1 Projektabschluss

Kriterien für den erfolgreichen Projektabschluss.

11.2 Zukunftsperspektiven

Erweiterungsmöglichkeiten der App (z.B. zusätzliche Funktionen oder Plattformen).