Projekthandbuch: PhileTipTip

Matthias Hochmuth

October 22, 2024

Contents

1 Einleitung

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.

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 Projektteam

Auflistung der Teammitglieder und deren Rollen (Projektmanager, Entwickler, Tester, etc.).

2.2 Verantwortlichkeiten

Aufgabenverteilung, Kommunikationswege und Entscheidungsfindung.

2.3 Zeitplan

Überblick über Meilensteine und Deadlines.

3 Anforderungsanalyse

3.1 Funktionale Anforderungen

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

3.2 Nicht-funktionale Anforderungen

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

- Benutzerfreundlichkeit
- Performance und Sicherheit
- 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

3.3 Technische Anforderungen

- Android-Kompatibilität.
- Daten On-Premises, keine Cloud

3.4 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.

Table 1: Ausgewählte User Stories

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

5 Technische Architektur

5.1 Systemarchitektur

5.1.1 Client-Server-Modell

Die App PhileTipTip basiert 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) ist 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.

5.1.2 Integration der Datenbank

Die MySQL-Datenbank ist fest in das Backend integriert und dient als zentrale Speicherinstanz für die erfassten Meldungen. 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.

5.1.3 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.

5.2 Technologien

5.2.1 Android Studio und Java

Die Entwicklung der App erfolgt in Android Studio, der offiziellen integrierten Entwicklungsumgebung (IDE) für Android. Android Studio bietet umfangreiche Tools für die Entwicklung, das Debugging und die Analyse der App-Performance. Als Programmiersprache wird Java verwendet, eine bewährte Sprache für die Android-Entwicklung. Java bietet eine große Entwickler-Community und umfangreiche Bibliotheken, die den Entwicklungsprozess beschleunigen und eine stabile, skalierbare App gewährleisten.

5.2.2 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.

5.3 Schnittstellen

5.3.1 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.

5.3.2 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.

6 Implementierung

6.1 Quellcodeverwaltung und Projektorganisation

6.1.1 Versionsverwaltung mit Git

Für die Versionskontrolle des Projekts kommt Git zum Einsatz. Git ermöglicht es dem Entwicklerteam, Änderungen am Code effizient nachzuverfolgen, neue Features in separaten Branches zu entwickeln und verschiedene Versionen der App zu verwalten. Durch die Verwendung von Git wird die Zusammenarbeit im Team vereinfacht und mögliche Konflikte bei der Integration von Codeänderungen minimiert.

6.1.2 Projektmanagement mit Jira

Die Projektplanung und -verfolgung erfolgt über Jira, ein führendes Tool für das agile Projektmanagement. Mit Jira können Aufgaben und Anforderungen in Form von Tickets angelegt, priorisiert und nachverfolgt werden. Das Tool bietet auch Funktionen zur Sprint-Planung, Fortschrittsüberwachung und Berichterstellung, was eine effiziente und transparente Verwaltung des gesamten Entwicklungsprozesses ermöglicht.

6.2 Codierstandards

6.3 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.

6.3.1 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.

6.3.2 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.

6.3.3 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.

6.4 Feature-Entwicklung

Schwerpunkte und iterative Entwicklung.

6.5 Tests

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

7 Qualitätssicherung

7.1 Testplan

Zeitplan für Tests und Fehlerbehebung.

7.2 Testumgebung

Beschreibung der Geräte und Android-Versionen.

7.3 Abnahmekriterien

Anforderungen für den Produktivstart.

8 Datenschutz und Sicherheit

8.1 Datenschutz

Umgang mit personenbezogenen Daten, Speicherung und Weiterleitung der Meldungen.

8.2 Sicherheitsrichtlinien

Zugriffskontrollen, Verschlüsselung von Daten, Backup-Strategien.

9 Deployment und Wartung

9.1 Veröffentlichung

App-Release im Google Play Store, Versionierung und Updates.

9.2 Wartungsplan

Fehlerbehebungen, Patches und langfristige Weiterentwicklung.

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).