

Projekt: Software Engineering

Projektdokumentation

DLMCSPSE01_D

Art der Arbeit:	Portfolio
Kursbezeichnung:	DLMCSPSE01_D
Studiengang:	FS MAINF-120 Fernstudium Master of Science Informatik 120 ECTS
Datum:	19.08.2024
Name:	Johannes Fell
Matrikelnummer:	92205755
Name Tutorin:	Markus Kleffmann

Inhalt

1.	Projektdokumentation	1
1.1.	Softwareprozess / Vorgehensmodell	1
1.2.	Technologien und Tools.....	2
1.3.	Design Pattern	5
1.3.1	MVC-Entwurfsmuster: Grundlagen und Vorteile	5
1.3.2	Der Nutzen von MVC in JavaFX-Anwendungen	5
1.3.3	Einsatz des erweiterten MVC-Patterns in der "PlantPal"-App	6
1.4.	Wichtige Implementierungsentscheidungen	8
1.5.	Struktur der Anwendung.....	10
1.5.1	Klassendiagramm der Abstrakten Systemarchitektur	10
1.5.2	Klassendiagramm der Steuerungs- und Geschäftslogikmodule.....	10
1.5.3	Klassendiagramm der Datenbank- und Modulmodule	11
1.5.4	Klassendiagramm der Geschäftslogik- und Datenbankmodulen.....	11
1.5.5	Gesamtdarstellung der Anwendung.....	11
1.6.	UML-Diagramme zur Prozessdarstellung.....	12
1.6.1	Aktivitätsdiagramm des Pflanzenprofil-Erstellungsprozesses	12
1.6.2	Sequenzdiagramm für den Foto-Upload und das Laden in der Slideshow.....	13
2.	Benutzeranleitung	14
2.1.	Installation.....	14
2.1.1	Installation über die .exe-Datei	14
2.1.2	Installation über die .jar-Datei.....	15
2.2.	Sicht „Pflanzen“.....	16
2.2.1	Pflanze hinzufügen.....	17
2.2.2	Pflanze aktualisieren	18
2.2.3	Pflanze löschen.....	19
2.2.4	Bild hochladen und in der Slideshow anzeigen	19
2.3.	Sicht „Pflege“	22
2.3.1	Pflegeaufgaben filtern	23
2.3.2	Pflegeaufgabe erledigen	23
2.3.3	Notizen verwalten.....	25
2.3.4	Pflegebenachrichtigungen	27
2.4.	Sicht „Wissensdatenbank“	29
2.5.	Sicht „Einstellungen“	31

Anhangsverzeichnis

Anhang 1: UML-Klassendiagramm der abstrakten Systemarchitektur	I
Anhang 2: UML-Klassendiagramm der Steuerungs- und Geschäftslogikmodule	II
Anhang 3: UML-Klassendiagramm der Datenbank- und Modelmodule.....	IV
Anhang 4: UML-Klassendiagramm der Geschäftslogik- und Datenbankmodule	V
Anhang 5: UML-Klassendiagramm Gesamtdarstellung	VI

Version

1.0	19.08.2024	Initiale Erstellung der Projektdokumentation
2.0	15.10.2024	<ul style="list-style-type: none">- Aktualisierung der Technologien und Tools- Hinzufügen von Kapitel 1.3 Design Pattern- Hinzufügen von Kapitel 1.4 Wichtige Implementierungsentscheidungen- Erweiterung/Verfeinerung des Kapitels 1.5 Struktur der Anwendung und Hinzufügen der UML-Diagramme im Anhang- Hinzufügen von Kapitel 1.6 UML-Diagramme zur Prozessdarstellung
3.0	18.10.2024	<ul style="list-style-type: none">- Erweiterung der Implementierungsentscheidungen (Kapitel 1.4) um Implementierungen für die Erstellung einer .exe Datei- Erstellung der Benutzeranleitung (Kapitel 2)

GitHub-Repository

https://github.com/JohannesFell/DLMCSPSE01_D-PlantPal

1. Projektdokumentation

1.1. Softwareprozess / Vorgehensmodell

Für die Entwicklung der "PlantPal"-App wurde das Kanban-Vorgehensmodell gewählt. Kanban ist ein flexibles und visuelles Managementsystem, das sich besonders gut für Softwareentwicklungsprojekte eignet, bei denen kontinuierliche Verbesserung, Transparenz und Anpassungsfähigkeit im Vordergrund stehen. Im Gegensatz zu anderen agilen Methoden wie Scrum, die in Iterationen arbeiten, konzentriert sich Kanban darauf, den gesamten Arbeitsfluss zu optimieren und Engpässe in Echtzeit zu identifizieren und zu beheben.

Kanban erlaubt es, Anforderungen und Aufgaben jederzeit zu priorisieren und neu zu bewerten, was besonders in einem Projekt wie "PlantPal" von Vorteil ist, wo sich Anforderungen während der Entwicklung ändern können. Durch den kontinuierlichen Fluss von Aufgaben ist es möglich, schnell auf neue Erkenntnisse oder Änderungen in den Projektanforderungen zu reagieren. Ein zentrales Element von Kanban ist das Kanban-Board, auf dem alle Aufgaben visualisiert werden. Dies ermöglicht eine klare Übersicht über den aktuellen Stand des Projekts und hilft, Engpässe schnell zu identifizieren und zu beheben.

Kanban fördert zudem eine Kultur der kontinuierlichen Verbesserung, bei der regelmäßig überprüft wird, wie der Entwicklungsprozess optimiert werden kann. Eine der Kernideen von Kanban ist die Begrenzung der gleichzeitig in Bearbeitung befindlichen Aufgaben. Dies verhindert, dass sich der Einzelentwickler verzettelt und zu viele Aufgaben parallel angeht, was oft zu Ineffizienz führt. Durch die Fokussierung auf eine begrenzte Anzahl von Aufgaben kann jede Aufgabe schneller und mit höherer Qualität abgeschlossen werden.

Da das "PlantPal"-Projekt von einem einzelnen Entwickler umgesetzt wird, ist Kanban besonders geeignet. Es erfordert keine umfangreichen Planungsmeetings oder komplexe Teamkoordination wie in Scrum. Stattdessen ermöglicht Kanban es, den Arbeitsaufwand kontinuierlich zu steuern und Prioritäten eigenständig anzupassen, was den Arbeitsfluss erleichtert und die Projektdurchführung effizienter gestaltet. Kanban bietet somit ein flexibles Vorgehensmodell, das speziell auf die Bedürfnisse des "PlantPal"-Projekts zugeschnitten ist.

1.2. Technologien und Tools

Programmiersprache: Java

Die Wahl von Java als Programmiersprache für die "PlantPal"-App basiert auf mehreren Überlegungen. Java ist eine etablierte, plattformunabhängige Sprache, die sich durch ihre Stabilität und Sicherheit auszeichnet. Java ermöglicht es, robuste und wartbare Anwendungen zu entwickeln, was für eine Desktop-Anwendung wie "PlantPal" entscheidend ist. Zudem bietet Java eine breite Palette von Bibliotheken und Frameworks, die die Entwicklung effizienter gestalten.

Entwicklungsumgebung: IntelliJ IDEA

IntelliJ IDEA ist eine integrierte Entwicklungsumgebung (IDE), die im Projekt eingesetzt wurde. Sie bietet Funktionen wie intelligente Code-Vervollständigung, Refactoring-Tools und eine Integration mit Build-Tools wie Maven oder Gradle, die den Entwicklungsprozess unterstützen.

Framework für GUI: JavaFX

JavaFX wird als Framework zur Entwicklung der grafischen Benutzeroberfläche (GUI) eingesetzt. JavaFX bietet eine moderne, benutzerfreundliche Oberfläche und unterstützt eine Vielzahl von Komponenten, die für die "PlantPal"-App relevant sind, wie z. B. Tabellen und Buttons. JavaFX ermöglicht es zudem, ansprechende visuelle Effekte und Animationen zu integrieren, was die Benutzererfahrung verbessert.

- **ControlsFX (v11.1.0):** Ein Erweiterungspaket für JavaFX, das zusätzliche UI-Komponenten bietet, die in der Entwicklung der GUI hilfreich sind und die Benutzerfreundlichkeit erhöhen.

Datenbank: SQLite

Für die lokale Speicherung der Pflanzendaten und Fotos wird SQLite verwendet. SQLite ist eine leichtgewichtige, serverlose SQL-Datenbank, die sich ideal für Desktop-Anwendungen eignet. Sie erfordert keine separate Installation oder Verwaltung, was die Komplexität der Anwendung reduziert und gleichzeitig eine zuverlässige Datenpersistenz gewährleistet.

- **SQLite JDBC (v3.46.1.0):** Diese JDBC-Treiber ermöglichen den Zugriff auf die SQLite-Datenbank von der Java-Anwendung aus, um alle Pflanzendaten effizient zu speichern und abzurufen.

E-Mail-Benachrichtigungen: MailJet

MailJet wird für die E-Mail-Benachrichtigungen genutzt, da es eine zentrale Plattform bietet, über die alle notwendigen Parameter konfiguriert werden können, wodurch der E-Mail-Versand einfach und benutzerfreundlich ist.

- **MailJet Client (v5.2.1):** Wird für die Integration von E-Mail-Benachrichtigungen verwendet. Es erleichtert das Versenden von Erinnerungen an Pflegeaktivitäten, ohne die Komplexität der manuellen SMTP-Konfiguration.

JSON-Verarbeitung: Gson

- **Gson (v2.8.9):** Gson ist eine Java-Bibliothek, die für die Umwandlung von Java-Objekten in JSON und umgekehrt verwendet wird. Dies ist besonders hilfreich bei der Verarbeitung und Speicherung von API-Daten oder anderen strukturierten Datenformaten.

Verbindungs-Pooling: HikariCP

- **HikariCP (v5.1.0):** HikariCP wird als Verbindungspool verwendet, um die Datenbankverbindungen effizient zu verwalten. Dies verbessert die Performance der Anwendung, insbesondere beim Zugriff auf die SQLite-Datenbank, da häufige Verbindungsanforderungen minimiert werden.

Logging: SLF4J

- **SLF4J API (v2.0.7) und SLF4J Simple (v2.0.7):** Diese Bibliotheken werden zur Protokollierung verwendet, um Fehler und Ereignisse innerhalb der Anwendung zu erfassen. SLF4J stellt eine allgemeine Schnittstelle bereit, die mit verschiedenen Logging-Implementierungen arbeiten kann.

Build- und Abhängigkeitsmanagement: Maven

Maven wird als Build-Tool und für das Abhängigkeitsmanagement verwendet. Maven ermöglicht eine einfache Verwaltung von Third-Party-Libraries und sorgt dafür, dass alle notwendigen Bibliotheken und Frameworks automatisch heruntergeladen und in das Projekt eingebunden werden. Zudem erleichtert Maven das Erstellen von Builds und das Testen der Anwendung.

Versionskontrolle: Git & GitHub

Zur Versionskontrolle wird Git in Kombination mit GitHub verwendet. Git bietet eine robuste und weit verbreitete Lösung für die Versionsverwaltung, die es ermöglicht, den Entwicklungsverlauf nachzuvollziehen und bei Bedarf auf frühere Versionen der Anwendung

zurückzugreifen. GitHub dient als zentrales Repository für den Quellcode und ermöglicht es, den Code sicher zu speichern und bei Bedarf mit anderen Entwicklern zu teilen.

Unit-Tests: JUnit

Zur Sicherstellung der Code-Qualität werden Unit-Tests mit JUnit durchgeführt. JUnit ist ein etabliertes Framework für das Testen von Java-Anwendungen und ermöglicht es, einzelne Komponenten der Anwendung isoliert zu testen. Durch den Einsatz von JUnit können Fehler frühzeitig erkannt und behoben werden, was die Zuverlässigkeit der Anwendung erhöht.

- **JUnit Jupiter API & Engine (v5.8.1):** Diese JUnit-Bibliotheken werden verwendet, um Unit-Tests für die Anwendung zu schreiben und auszuführen, sodass die Funktionsweise einzelner Komponenten zuverlässig überprüft werden kann.

1.3. Design Pattern

Die Struktur der "PlantPal"-Anwendung folgt einem erweiterten MVC-Architekturmuster (Model-View-Controller). Dieses Entwurfsmuster hilft dabei, die unterschiedlichen Verantwortlichkeiten der Anwendung zu trennen und sorgt somit für eine bessere Wartbarkeit, Wiederverwendbarkeit und Erweiterbarkeit der Software. Im Folgenden wird das MVC-Pattern erklärt, der Einsatz in JavaFX begründet und die spezifische Anwendung in der "PlantPal"-App beschrieben.

1.3.1 MVC-Entwurfsmuster: Grundlagen und Vorteile

Das Model-View-Controller (MVC) ist ein bewährtes Architekturkonzept, welches darauf abzielt, die Software in drei Hauptkomponenten zu unterteilen:

- **Model:** Das Model repräsentiert die Daten der Anwendung sowie die zugehörigen Logiken zur Datenmanipulation. Es ist dafür verantwortlich, den aktuellen Zustand der Anwendung zu speichern und zu ändern, wenn die Daten sich verändern.
- **View:** Die View stellt die Benutzeroberfläche dar und ist dafür verantwortlich, die Daten des Models grafisch aufzubereiten. Sie interagiert direkt mit dem Benutzer und zeigt die entsprechenden Informationen an.
- **Controller:** Der Controller dient als Bindeglied zwischen Model und View. Er empfängt die Benutzereingaben, verarbeitet diese und steuert die entsprechenden Änderungen am Model oder Updates der View.

Durch diese Trennung werden die Verantwortlichkeiten klar verteilt, was zu einer besseren Struktur der Anwendung führt. Dies ermöglicht eine einfachere Wartung und Erweiterung, da beispielsweise die Benutzeroberfläche (View) angepasst werden kann, ohne die Logik des Models oder des Controllers zu beeinflussen.

1.3.2 Der Nutzen von MVC in JavaFX-Anwendungen

JavaFX ist ein modernes Framework für die Entwicklung von grafischen Benutzeroberflächen in Java-Anwendungen und eignet sich hervorragend für die Anwendung des MVC-Patterns. Ein typisches JavaFX-Projekt ist modular aufgebaut, wodurch die Trennung der Verantwortlichkeiten bereits natürlich unterstützt wird.

- **Einfache Wartbarkeit und Testbarkeit:** Durch die Trennung von Daten, Logik und Präsentation lässt sich jede dieser Komponenten separat testen und weiterentwickeln. In JavaFX kann das Model unabhängig von der Benutzeroberfläche getestet werden,

während die View mithilfe von FXML-Dateien erstellt wird, die die visuelle Darstellung klar definieren.

- **Erweiterbarkeit:** MVC fördert die Modularität, sodass einzelne Komponenten leicht ausgetauscht oder erweitert werden können. Beispielsweise könnten neue Views hinzugefügt werden, ohne dass die zugrunde liegende Logik geändert werden muss.
- **Parallele Entwicklung:** Durch die klare Trennung von Aufgaben kann die Entwicklung der Oberfläche und der Logik parallel erfolgen, was in großen Teams die Produktivität steigert. Obwohl "PlantPal" von einem Einzelentwickler umgesetzt wird, profitiert auch dieser von einer klaren Struktur und Übersichtlichkeit.

1.3.3 Einsatz des erweiterten MVC-Patterns in der "PlantPal"-App

Die "PlantPal"-Anwendung nutzt eine erweiterte Form des MVC-Patterns, bei der zusätzlich ein Service Layer für die Geschäftslogik eingesetzt wird, um eine klare Trennung der Verantwortlichkeiten sicherzustellen. Die Struktur der Anwendung umfasst:

- **Model:** Die Klassen im model-Paket, wie PflanzenPflege_Model, Einstellungen_Model und PflanzenProfile_Model, repräsentieren das Datenmodell der Anwendung. Diese Klassen sind dafür zuständig, die Pflanzendaten, Pflegeaufgaben und Benutzereinstellungen zu speichern und zu verwalten. Die Modelle enthalten die wesentlichen Attribute und Methoden zur Verwaltung des Zustands der Anwendung, wie z.B. task_id, due_date oder completed für Pflegeaufgaben.
- **View:** Die grafische Benutzeroberfläche der Anwendung wird mithilfe von JavaFX und FXML implementiert. Die View ist für die Darstellung der Daten und die Benutzerinteraktion zuständig. JavaFX ermöglicht es, diese visuelle Darstellung mit FXML-Dateien klar zu definieren, wodurch eine einfache Anpassung der Benutzeroberfläche möglich ist. Die Views stellen sicher, dass die Benutzerfreundlichkeit und die visuelle Aufbereitung den Anforderungen entsprechen.
- **Controller:** Die Controller-Klassen im app-Paket agieren als Vermittler zwischen View und Model. Beispielsweise verarbeitet der PflanzenPflegeController die Pflegeaufgaben, indem er die Benutzereingaben aus der View entgegennimmt und die Daten im Model ändert. Gleichzeitig sorgt er dafür, dass Änderungen im Model korrekt in der View dargestellt werden. Controller-Klassen wie MainScreenController, PflanzenProfileController und PflanzenPflegeController sind dafür verantwortlich, Benutzereingaben zu verarbeiten, mit den entsprechenden Services zu interagieren und die Views entsprechend zu aktualisieren.

- **Service Layer:** Die Geschäftslogik wird in den Klassen des logic-Pakets abgebildet. Dieser Layer erweitert das klassische MVC-Pattern, indem er eine separate Schicht zur Verwaltung der Geschäftsprozesse bereitstellt. Der PflegeAufgabenService ist beispielsweise für die Verwaltung der Pflegeaufgaben zuständig, indem er Methoden zum Laden, Erstellen und Markieren von Aufgaben als erledigt bereitstellt. Die Services stellen sicher, dass die Geschäftsregeln der Anwendung an einem zentralen Ort implementiert werden, was die Wiederverwendbarkeit und Testbarkeit der Logik erleichtert. Die Service-Klassen, wie PflegeAufgabenService, PflanzenProfileService oder EmailService, interagieren mit den Repositories im database-Paket, um Daten zu speichern und abzurufen.

Durch die Anwendung dieses erweiterten MVC-Patterns wird die Wartbarkeit der Anwendung erhöht, da die verschiedenen Komponenten – das Model, die Benutzeroberfläche, die Controller und die Logik – voneinander entkoppelt sind. Diese Trennung ermöglicht es, Erweiterungen oder Anpassungen an einer der Komponenten vorzunehmen, ohne die anderen zu beeinflussen. Dies ist besonders für die langfristige Wartung und Weiterentwicklung der Software von Vorteil. Zudem wird eine klare Struktur geschaffen, die die Entwicklung übersichtlich und effizient macht. Der Einsatz eines dedizierten Service Layers erlaubt es, die Geschäftslogik aus den Controllern herauszulösen, was zu einer sauberen Trennung von Benutzerinteraktionen und Geschäftsregeln führt.

1.4. Wichtige Implementierungsentscheidungen

Neben der Anwendung des erweiterten MVC-Patterns sind weitere wichtige Design- und Implementierungsentscheidungen im "PlantPal"-Projekt getroffen worden, die hier erläutert werden:

- **Datenbankwahl und Datenmanagement:** Die Entscheidung fiel auf SQLite als Datenbanktechnologie, weil sie sich gut für Desktop-Anwendungen eignet. SQLite ist leichtgewichtig und serverlos, wodurch keine zusätzliche Infrastruktur notwendig ist. Das Repository Pattern wird verwendet, um eine klare Trennung zwischen der Datenzugriffsschicht und der Geschäftslogik zu gewährleisten, was die Wartbarkeit erhöht.
- **Einsatz von Service Layern:** Durch den Einsatz von Service Layern wie dem PflegeAufgabenService wird die Geschäftslogik von der Controller-Logik getrennt. Dadurch bleiben die Controller schlank und fokussiert auf die Benutzerinteraktion, während die Geschäftsregeln in den Services gekapselt sind.
- **JavaFX als GUI Framework:** FXML wird verwendet, um die Benutzeroberfläche zu definieren. Das erleichtert die Wartung und Anpassung des Layouts, da die GUI-Komponenten von der Logik getrennt sind. Die Nutzung von JavaFX-Komponenten und @FXML-Annotationen ermöglicht eine saubere Bindung zwischen der Benutzeroberfläche und der Controller-Logik.
- **Erstellung von .exe für einfachere Auslieferung:**
 - **jlink und jpackage** wurden verwendet, um ein benutzerdefiniertes Java Runtime Environment (JRE) zu erstellen und die Anwendung als .exe bereitzustellen. Dies erleichtert die Verteilung, da keine zusätzliche JRE-Installation erforderlich ist.
 - **Ergänzungen zur pom.xml:** Für die Erstellung der .exe wurden zusätzliche Maven-Plugins und Konfigurationen hinzugefügt, einschließlich **javafx-maven-plugin** für die JavaFX-Integration und **maven-shade-plugin** zur Erzeugung eines ausführbaren JARs. Durch das Erstellen eines nativen Installationspakets mit jpackage können Benutzer die Anwendung durch einfaches Ausführen der .exe nutzen, ohne zusätzliche Konfigurationen oder Abhängigkeiten installieren zu müssen.
- **E-Mail-Versand über MailJet:** Für die E-Mail-Benachrichtigungen wird MailJet genutzt, da es eine einfache und benutzerfreundliche Lösung für den Versand von Erinnerungen darstellt. Der EmailService übernimmt die Integration und Kommunikation mit der MailJet-API.

- **Multithreading für zeitgesteuerte Aufgaben:** Der Java ScheduledExecutorService wird verwendet, um wiederkehrende Aufgaben wie Pflegebenachrichtigungen im Hintergrund zu erledigen, ohne die Hauptanwendung zu blockieren. Dadurch bleibt die Anwendung responsiv.
- **JUnit Jupiter API & Engine:** Diese Bibliotheken werden verwendet, um Unit-Tests zu schreiben und auszuführen. Sie sind entscheidend, um die Qualität und Stabilität des Codes sicherzustellen, indem sie ermöglichen, die einzelnen Komponenten isoliert zu testen.
- **Gson:** Gson wird verwendet, um Java-Objekte in JSON zu konvertieren und umgekehrt. Diese Bibliothek ist hilfreich, wenn Daten zwischen der Anwendung und externen APIs ausgetauscht werden, da JSON ein weit verbreitetes Format für die Datenübertragung ist.
- **HikariCP:** HikariCP ist ein Connection-Pooling-Framework für JDBC. Es wird verwendet, um die Performance der Anwendung zu verbessern, indem Datenbankverbindungen effizient verwaltet werden. Dies reduziert die Latenzzeiten beim Zugriff auf die SQLite-Datenbank und stellt sicher, dass die Anwendung auch bei hoher Last performant bleibt.

1.5. Struktur der Anwendung

Die Struktur der "PlantPal"-App folgt einem modularen Ansatz, der auf einem erweiterten MVC-Architekturmuster (Model-View-Controller) basiert. Ziel ist es, Verantwortlichkeiten klar zu trennen, die Wartbarkeit und Erweiterbarkeit zu gewährleisten und den Code wiederverwendbar zu machen. Die Anwendung besteht aus mehreren Modulen, die in verschiedenen Paketen organisiert sind, um spezifische Aufgaben zu übernehmen. Die UML-Klassendiagramme bieten eine detaillierte Übersicht über die einzelnen Komponenten und deren Interaktionen. Die folgenden Abschnitte beschreiben die verschiedenen Pakete und ihre Beziehungen zueinander.

1.5.1 Klassendiagramm der Abstrakten Systemarchitektur

Das UML-Diagramm in Anhang 1 zur Systemarchitektur gibt einen abstrakten Überblick über die grundlegende Struktur der "PlantPal"-App. Es zeigt die Hauptmodule, einschließlich der Steuerung (app), der Geschäftslogik (logic), der Datenbank (database), der Modelle (model) und der unterstützenden Dienste (utils). Die Steuerungsschicht (app) interagiert direkt mit der Geschäftslogikschicht (logic) und greift auf die Datenbankschicht (database) zu, während die Geschäftslogik auf die Modelle (model) und Hilfsfunktionen (utils) zugreift. Diese lose Kopplung zwischen den Modulen ermöglicht eine klare Trennung der Verantwortlichkeiten und fördert eine hohe Wartbarkeit.

1.5.2 Klassendiagramm der Steuerungs- und Geschäftslogikmodule

Das UML-Klassendiagramm der Steuerungs- und Geschäftslogikmodule (Anhang 2) zeigt die detaillierte Struktur der Controller-Klassen im "app"-Paket und deren Beziehung zur Geschäftslogik im "logic"-Paket. Die Controller-Klassen, wie der BenachrichtigungsController, MainScreenController oder PflanzenPflegeController, sind für die Verarbeitung der Benutzereingaben und die Steuerung der Benutzeroberfläche zuständig. Sie agieren als Vermittler zwischen der grafischen Benutzerschnittstelle (GUI) und der Geschäftslogik, um Benutzeraktionen in Geschäftsvorgänge zu übersetzen. Diese lose Kopplung zwischen den Controllern und der Logik sorgt für eine effiziente Trennung der Verantwortlichkeiten und erleichtert die Erweiterbarkeit.

1.5.3 Klassendiagramm der Datenbank- und Modelmodule

Das UML-Klassendiagramm der Datenbank- und Modelmodule (Anhang 3) beschreibt die Architektur der Klassen im "database"-Paket und deren Beziehungen zu den Modellen im "model"-Paket. Die Klassen im "database"-Paket, wie SQLiteDB und PlantProfileRepository, sind für die Datenhaltung und die Kommunikation mit der SQLite-Datenbank zuständig. Das Repository-Entwurfsmuster wird verwendet, um eine klare Trennung zwischen der Datenzugriffsschicht und der Geschäftslogik zu gewährleisten. Die Repositories, wie PlantProfileRepository oder CareTaskRepository, führen CRUD-Operationen (Create, Read, Update, Delete) aus und interagieren mit den Modellen, wie PflanzenPflege_Model oder PflanzenProfile_Model, um die Daten in einer für die Geschäftslogik geeigneten Form zu liefern. Diese Beziehungen verdeutlichen, wie die Datenbank- und Repository-Module die Geschäftslogik und das Modell miteinander verbinden.

1.5.4 Klassendiagramm der Geschäftslogik- und Datenbankmodulen

Das UML-Klassendiagramm der Geschäftslogik (Anhang 4) beschreibt die Geschäftslogikmodule im "logic"-Paket und zeigt deren Interaktionen mit der Datenbank- und Repository-Schicht. Die Services, wie PflanzenProfileService, EmailService, PflegeAufgabenService und BenachrichtigungsService, führen die wesentlichen Geschäftsprozesse der Anwendung durch. Sie greifen auf die Repository-Klassen aus dem "database"-Paket zu, um Daten zu speichern oder abzurufen. Zum Beispiel nutzt der PflegeAufgabenService die CareTaskRepository-Klasse, um Pflegeaufgaben zu verwalten, während der ImageService für die Speicherung und Verwaltung von Bildern verantwortlich ist. Diese Services werden von den Controllern aufgerufen und ermöglichen eine zentrale Umsetzung der Geschäftslogik, was zu einer modularen und wiederverwendbaren Codebasis führt.

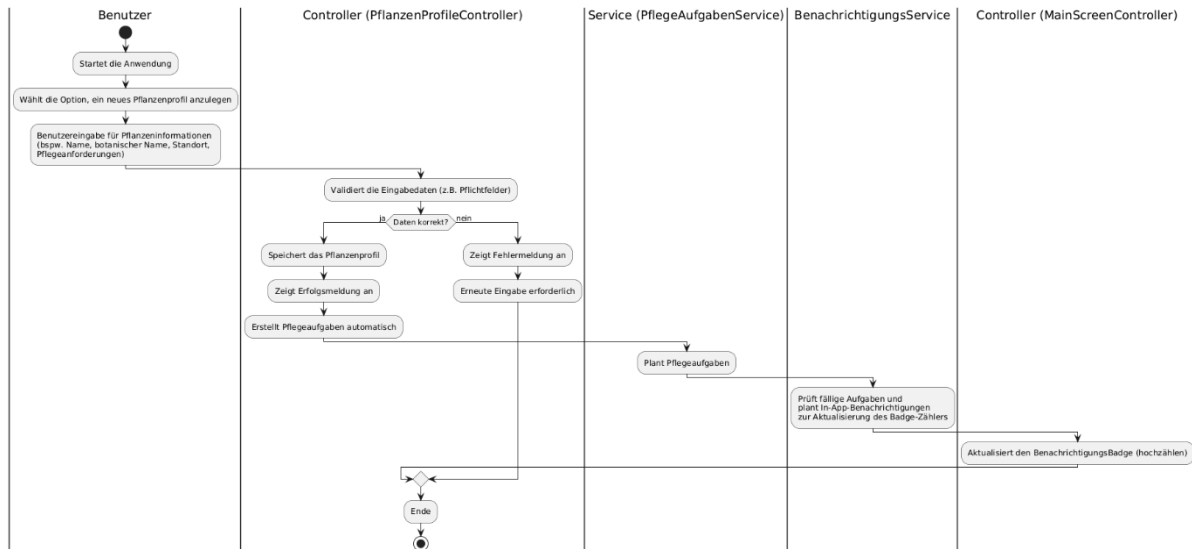
1.5.5 Gesamtdarstellung der Anwendung

Das abschließende UML-Diagramm (Anhang 5) bietet eine vollständige Gesamtdarstellung der Anwendung, indem es alle zuvor beschriebenen Komponenten zusammenführt. Es zeigt die Beziehungen zwischen den Controllern (app), den Services (logic), der Datenbank (database), den Modellen (model) und den Hilfsklassen (utils). Dieses Diagramm verdeutlicht, wie die verschiedenen Module zusammenarbeiten, um die Gesamtfunktionalität der "PlantPal"-App bereitzustellen. Die klare Trennung der einzelnen Schichten und die lose Kopplung fördern eine hohe Wartbarkeit und vereinfachen zukünftige Erweiterungen der Anwendung.

1.6. UML-Diagramme zur Prozessdarstellung

1.6.1 Aktivitätsdiagramm des Pflanzenprofil-Erstellungsprozesses

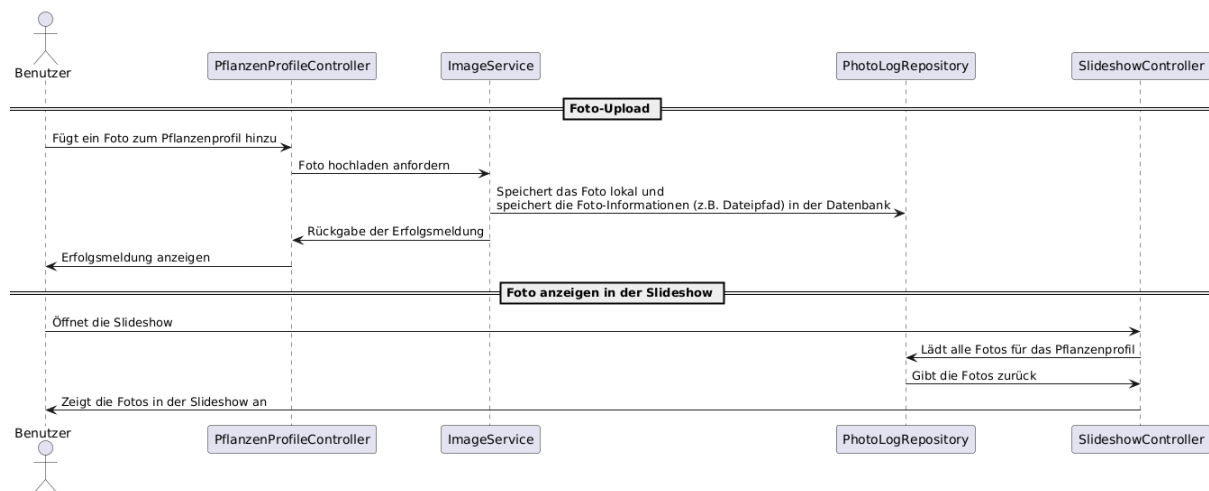
Diese UML-Aktivitätsdiagramm beschreibt den gesamten Ablauf vom Start der Anwendung, über die Eingabe und Validierung der Pflanzeninformationen, bis hin zur automatischen Planung der Pflegeaufgaben und der Aktualisierung des Benachrichtigungs-Badges.



1.6.2 Sequenzdiagramm für den Foto-Upload und das Laden in der Slideshow

Das Sequenzdiagramm zeigt den Prozess des Foto-Uploads und des anschließenden Ladens und Anzeigens der Fotos in der Slideshow. Der erste Teil beschreibt, wie der Benutzer ein Foto zum Pflanzenprofil hinzufügt. Das Foto wird durch den „ImageService“ hochgeladen und im „PhotoLogRepository“ gespeichert. Anschließend wird eine Erfolgsmeldung an den Benutzer zurückgegeben.

Der zweite Teil beschreibt den Vorgang des Ladens der Bilder, wenn der Benutzer die Slideshow öffnet. Der „SlideshowController“ lädt die Bilder aus dem „PhotoLogRepository“ und zeigt sie dem Benutzer in einer Slideshow an.



2. Benutzeranleitung

PlantPal ist eine Desktop-App, die entwickelt wurde, um Pflanzenbesitzern, Gartenliebhabern und Hobbygärtnern die Pflege ihrer Zimmerpflanzen zu erleichtern. Mit einer Vielzahl nützlicher Funktionen, wie der Verwaltung von Pflanzenprofilen, die Anzeige und das Versenden von Pflegeerinnerungen und einer integrierten Wissensdatenbank, bietet PlantPal eine umfassende Lösung, um Ihre Pflanzen gesund zu halten und ihr Wachstum zu dokumentieren.

Diese Anleitung soll Ihnen helfen, sich mit den Funktionen der PlantPal-App vertraut zu machen und die Bedienung Schritt für Schritt zu erklären. Von der Installation über das Hinzufügen neuer Pflanzen bis hin zur Nutzung der Pflegeerinnerungen finden Sie hier alle Informationen, die Sie benötigen, um die App optimal zu nutzen. PlantPal ist intuitiv gestaltet, sodass Sie die Pflege Ihrer Pflanzen einfach und effizient organisieren können.

Die folgenden Kapitel bieten eine detaillierte Erläuterung der verschiedenen Ansichten und Prozesse, die Ihnen zur Verfügung stehen, um Ihre Pflanzen optimal zu pflegen.

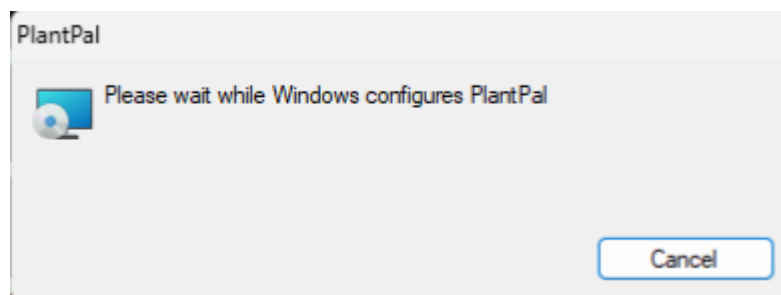
2.1. Installation

Um die PlantPal-Anwendung zu installieren, können Sie entweder die ausführbare .exe-Datei oder die .jar-Datei verwenden, die auf GitHub verfügbar sind:

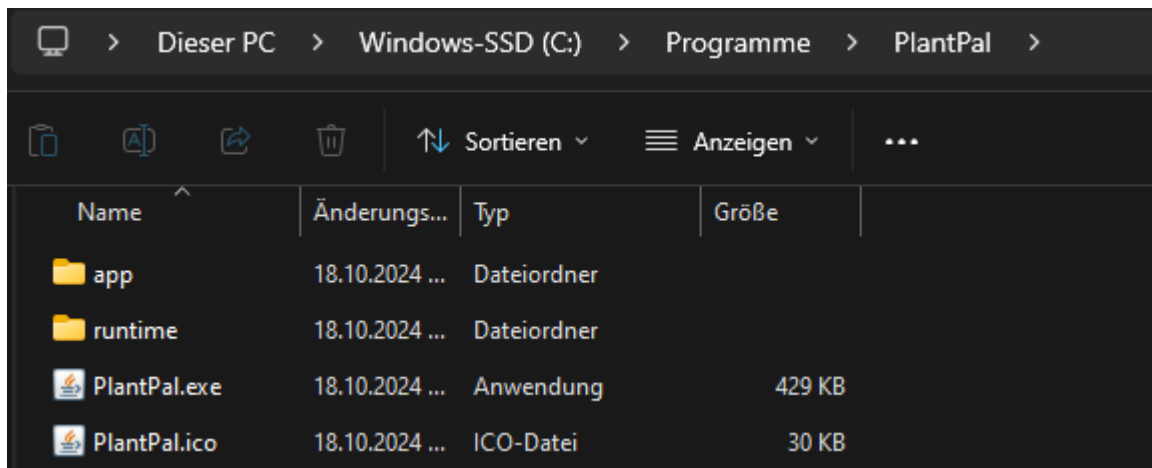
https://github.com/JohannesFell/DLMCSPSE01_D-PlantPal

2.1.1 Installation über die .exe-Datei

- Laden Sie die .exe-Datei (Installer: PlantPal-1.0.exe) von GitHub herunter.
- Führen Sie die Installation durch.



- Das Programm sollte nun unter Ihren Programmen (C:\Program Files\PlantPal) installiert worden sein und folgende Dateien beinhalten:



- Führen Sie die PlantPal.exe-Datei als Administrator aus. Dies ist notwendig, um sicherzustellen, dass die Anwendung über die erforderlichen Rechte verfügt, um die Datenbank zu beschreiben und alle Funktionen korrekt auszuführen.

2.1.2 Installation über die .jar-Datei

- Laden Sie die .jar-Datei von GitHub herunter.
- Um die .jar-Datei auszuführen, müssen Sie das folgende Kommando verwenden:

```
java --module-path "C:\path\to\javafx\lib" --add-modules javafx.controls,javafx.fxml,javafx.web -jar PlantPal-1.0-SNAPSHOT.jar
```
- Dies ist erforderlich, weil die Anwendung JavaFX-Module verwendet, die nicht standardmäßig im JDK enthalten sind. Beim Start der .jar-Datei müssen Sie daher explizit den Pfad zu den JavaFX-Bibliotheken (module-path) angeben und die benötigten Module hinzufügen (--add-modules). Dies stellt sicher, dass die JavaFX-Bibliotheken korrekt geladen werden und Ihre Anwendung ausgeführt werden kann. Andernfalls kann es zu Fehlern wegen fehlender Module oder Klassen kommen.

Nachdem die Installation abgeschlossen ist, können Sie mit der Nutzung von PlantPal beginnen.

2.2. Sicht „Pflanzen“

Die Benutzeroberfläche zum Verwalten von Pflanzenprofilen ist einfach und intuitiv gestaltet, sodass Sie problemlos neue Pflanzenprofile erstellen können. Die Hauptkomponente der Ansicht ist eine Tabelle, die Ihnen eine Übersicht über alle bereits hinzugefügten Pflanzen gibt. Zusätzlich gibt es ein Formular, mit dem Sie die Details zu einer neuen Pflanze eintragen können.

1	Suchfeld	Mit dem Suchfeld können Sie nach spezifischen Pflanzen suchen, indem Sie den (botanischen) Namen oder den Standort eingeben
2	Tabelle der Pflanzenprofile	Die Tabelle zeigt wichtige Informationen wie den Namen, den botanischen Namen, den Standort, das letzte Gießen und Düngen sowie die jeweiligen Intervalle an. Dies ermöglicht eine schnelle Übersicht über den Zustand Ihrer Pflanzen
3	Name	Name der Pflanze
4	Name botanisch	Botanischer Name der Pflanze
5	Standort	Standort der Pflanze
6	Kaufdatum	Kaufdatum der Pflanze (dd.MM.yyyy)
7	Letztes Gießen	Zeigt an, wann die Pflanze das letzte Mal gegossen wurde.

8	Letztes Düngen	Zeigt an, wann die Pflanze das letzte Mal gedüngt wurde.
9	Intervall Gießen	Zeigt an, in welchem zeitlichen Abstand gegossen werden soll (in Tagen)
10	Intervall Düngen	Zeigt an, in welchem zeitlichen Abstand gedüngt werden soll (in Tagen)
11	Import	Es öffnet sich der Dateexplorer zum Upload von Bildern (.png, .jpg, .jpeg)
12	Show	Anzeige einer Slideshow mit den bereits hochgeladenen Bildern
13	Bild	Vorschaubild (falls vorhanden – sonst <i>default</i> Bild)
14	Leeren	Leert das Formular
15	Löschen	Löschen des ausgewählten Pflanzenprofils (nach Bestätigung in neuem Dialog)
16	Aktualisieren	Aktualisieren des ausgewählten Pflanzenprofils
17	Hinzufügen	Hinzufügen eines neuen Pflanzenprofils

2.2.1 Pflanze hinzufügen

Mit dem Button „Hinzufügen“ können Sie eine neue Pflanze zur Datenbank hinzufügen.

Füllen Sie die entsprechenden Felder wie Name, botanischer Name, Standort, Kaufdatum sowie Gieß- und Düngeintervalle aus und klicken Sie auf „Hinzufügen“.

Die Felder letztes Gießen und letztes Düngen können beim Anlegen einer neuen Pflanze nicht verwaltet werden. Hier wird automatisch das aktuelle Datum eingefügt.

Die Pflanze wird dann in der Tabelle der Pflanzenprofile angezeigt. Wenn nicht alle Pflichtfelder ausgefüllt sind oder Datumsformate nicht stimmen sollten, wird eine Benachrichtigung angezeigt, um sicherzustellen, dass alle erforderlichen Informationen eingegeben werden.

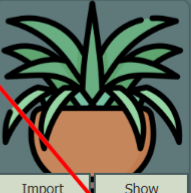
Gleich nach dem Anlegen einer neuen Pflanze werden automatisch die dazugehörigen Pflegeaufgaben in Abhängigkeit von den Gieß-/Düngeintervallen erstellt und ggfs. das Benachrichtigungs-Badge aktualisiert (Hierzu mehr in Kapitel 2.3.4).

Notification: 2

Suche

Name	Name botanisch	Standort	Letztes Gießen	Letztes Düngen	Intervall Gießen	Intervall Düngen	Kaufdatum
Fensterblatt	Monstera	Wohnzimmer	18.10.2024	18.10.2024	7 Tage	14 Tage	01.01.2024

Name: Fensterblatt
 Name botanisch: Monstera
 Standort: Wohnzimmer
 Kaufdatum: 01.01.2024
 Letztes Gießen: 18.10.2024
 Letztes Düngen: 18.10.2024
 Intervall Gießen: 7 Tage
 Intervall Düngen: 14 Tage

Leeren Löschen
 
 Import Show
 Hinzufügen

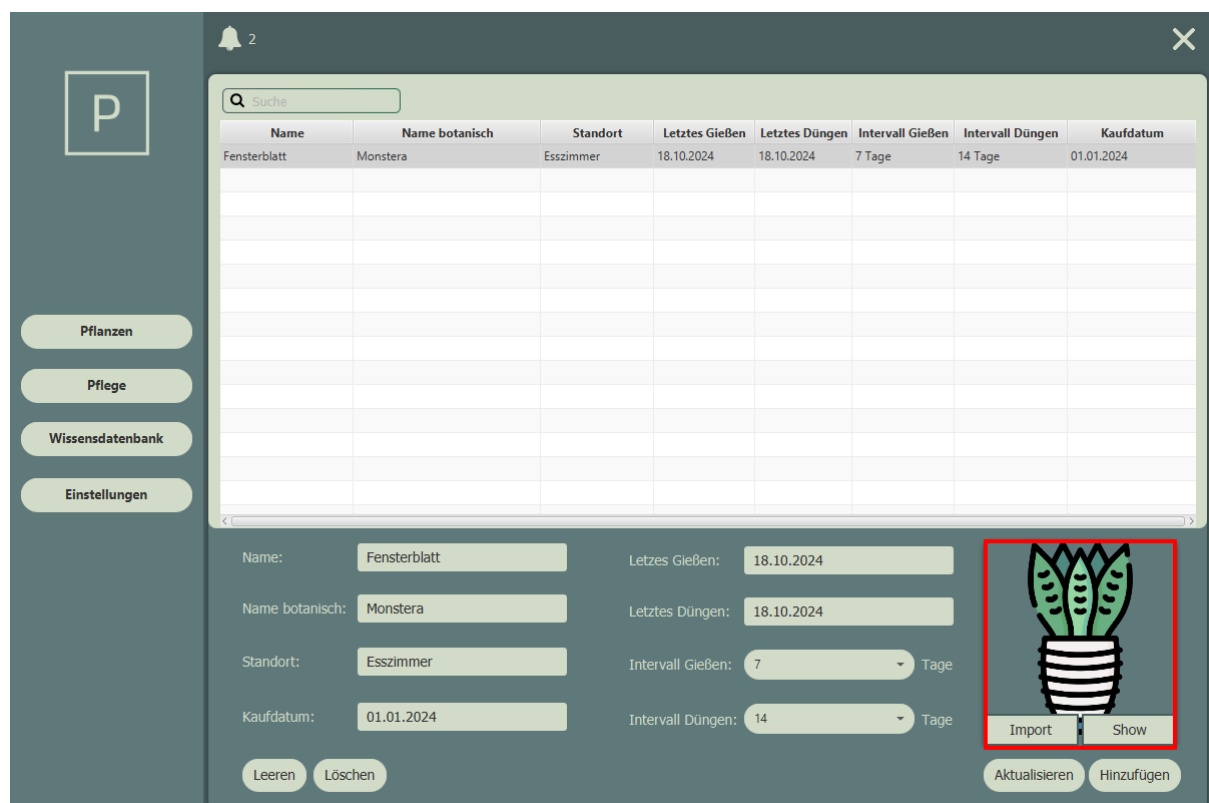
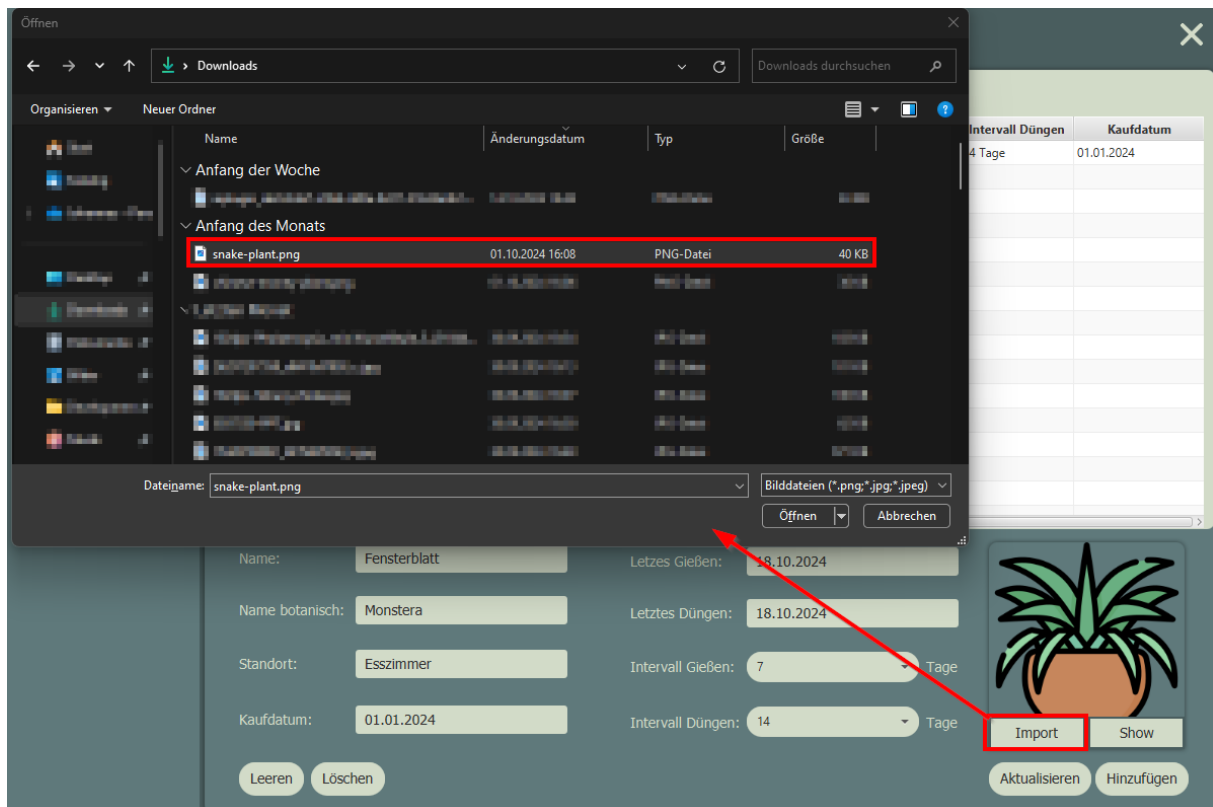
2.2.2 Pflanze aktualisieren

Wählen Sie eine Pflanze aus der Tabelle aus, um deren Details im Formular anzuzeigen. Ändern Sie die gewünschten Informationen und klicken Sie auf „Aktualisieren“, um die Änderungen zu speichern.

Dies kann beispielsweise dann sinnvoll sein, wenn Sie den Standort einer Pflanze ändern oder die Intervalle für Gießen und Düngen anpassen möchten.

Nach dem Aktualisieren werden die neuen Informationen sofort in der Tabelle angezeigt.

Zur besseren Nachverfolgung der Änderungen im Pflanzenprofil, werden alle Änderungen am (botanischen) Namen, Standort oder den Gieß-/Düngeintervallen in die Pflegehistorie eingetragen (Hierzu mehr in Kapitel 2.3.3).



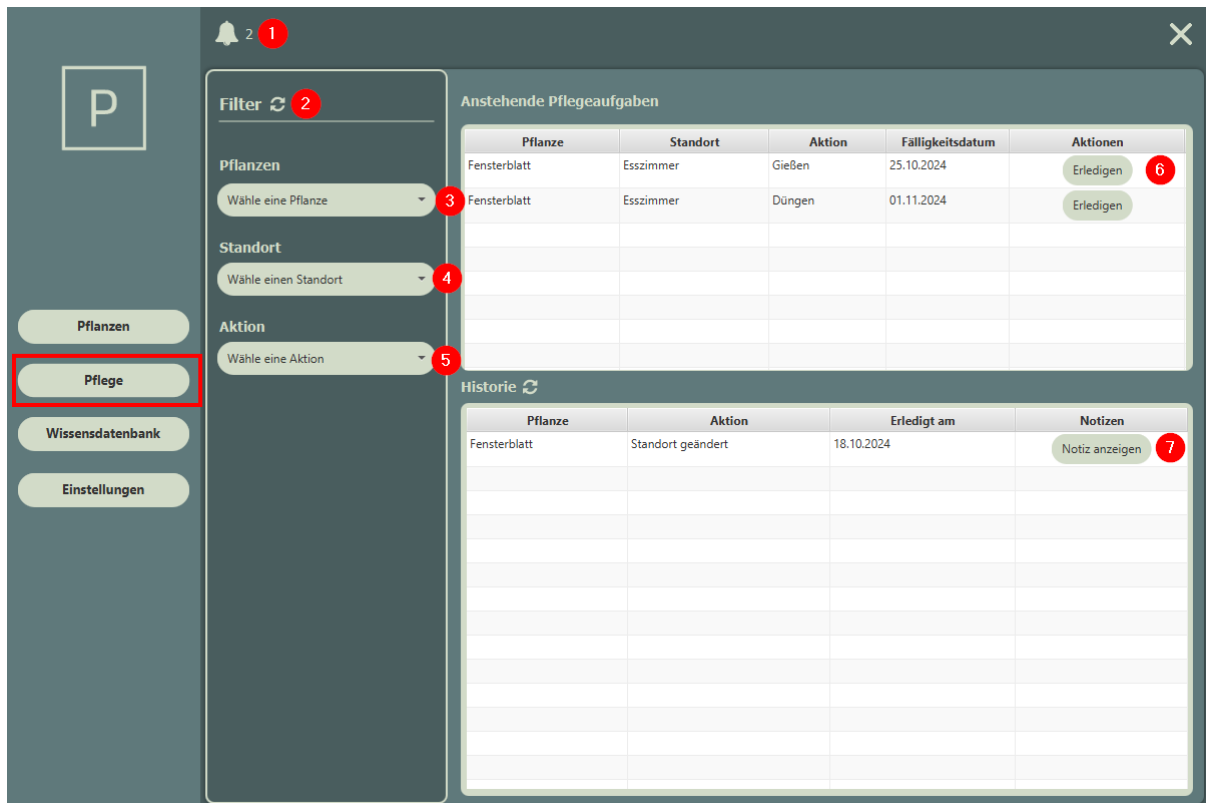
Um die hochgeladenen Bilder in einer Slideshow anzuzeigen, wählen Sie die entsprechende Pflanze aus der Tabelle aus und klicken Sie auf den Button „Show“. Die Slideshow zeigt Ihnen alle zu dieser Pflanze gespeicherten Bilder in chronologischer Reihenfolge. Dies ist besonders nützlich, um die Veränderungen Ihrer Pflanzen im Laufe der Zeit zu betrachten und

festzuhalten, wie gut sich Ihre Pflanzen entwickelt haben. Sie können durch die Bilder navigieren und so den Fortschritt und Veränderungen über verschiedene Zeiträume hinweg visualisieren.



2.3. Sicht „Pflege“

In der Ansicht „Pflege“ können Sie alle anstehenden Pflegeaufgaben für Ihre Pflanzen sowie eine Historie vergangener Aufgaben einsehen. Die Oberfläche bietet mehrere Funktionen, um die Pflege Ihrer Pflanzen optimal zu verwalten.



1	Benachrichtigungs-Badge	Das Benachrichtigungs-Badge zeigt (wenn in den Einstellungen aktiviert) die anstehenden Pflegeaufgaben im ausgewählten Zeitrahmen an.
2	Filter zurücksetzen	Dieser Button ist auf der Pflegemaske zwei Mal zu finden. Durch Betätigung des Buttons werden die ausgewählten Filter zurückgesetzt.
3	Name filtern	Filtern nach einem Pflanzennamen
4	Standort filtern	Filtern nach einem Standort
5	Aktionen filtern	Filtern nach einer Aktion (Gießen, Düngen, Änderungen im Profil)
6	Erledigen	Anstehende Aufgaben werden als erledigt gekennzeichnet und werden in der Historie angezeigt.
7	Notizen verwalten	Dynamischer Button zur Verwaltung von Notizen. Sollten keine Notizen zu einem Tabelleneintrag vorhanden sein, kann über „Notiz

		<p>hinzufügen“ eine neue Notiz hinzugefügt werden.</p> <p>Ist schon eine Notiz vorhanden, kann diese angezeigt und angepasst werden.</p>
--	--	--

2.3.1 Pflegeaufgaben filtern

Auf der linken Seite der Ansicht finden Sie Filteroptionen, mit denen Sie die Pflegeaufgaben gezielt eingrenzen können. Sie können nach einer bestimmten Pflanze, einem Standort oder einer bestimmten Pflegeaktion filtern, um schnell den Überblick über relevante Aufgaben zu behalten. Über den „Refresh“ Button (2) können alle gesetzten Filter wieder zurückgesetzt werden.

The screenshot shows the application interface with a sidebar on the left and a main content area on the right. The sidebar contains a 'Filter' section with three dropdown menus: 'Pflanzen', 'Standort', and 'Aktion'. The 'Aktion' dropdown is open, showing a list of actions: 'Gießen', 'Düngen', 'Name geändert', 'Standort geändert', 'Gießintervall geändert', and 'Düngeintervall geändert'. A red box highlights the 'Aktion' dropdown menu. A red arrow points from the 'Aktion' dropdown to the 'Aktion' column header in the 'Anstehende Pflegeaufgaben' table. Another red arrow points from the 'Aktion' dropdown to the 'Aktion' column header in the 'Historie' table. The 'Anstehende Pflegeaufgaben' table has columns: 'Pflanze', 'Standort', 'Aktion', 'Fälligkeitsdatum', and 'Aktionen'. It contains two rows of tasks. The 'Historie' table has columns: 'Pflanze', 'Aktion', 'Erledigt am', and 'Notizen'. It contains one row of history.

Pflanze	Standort	Aktion	Fälligkeitsdatum	Aktionen
Fensterblatt	Esszimmer	Gießen	25.10.2024	Erledigen
Fensterblatt	Esszimmer	Düngen	01.11.2024	Erledigen

Pflanze	Aktion	Erledigt am	Notizen
Fensterblatt	Standort geändert	18.10.2024	Notiz anzeigen

2.3.2 Pflegeaufgabe erledigen

Der Button „Erledigen“ ermöglicht es Ihnen, eine Pflegeaufgabe als abgeschlossen zu markieren. Wählen Sie dazu die entsprechende Aufgabe aus der Tabelle der anstehenden Pflegeaufgaben und klicken Sie auf „Erledigen“. Die Aufgabe wird dann in die Historie der Pflegeaufgaben verschoben und, mit dem Datum dem aktuellen Datum, als erledigt markiert. Dies stellt sicher, dass Ihre Pflegeübersicht stets aktuell ist und Sie immer wissen, welche Aufgaben noch offen sind.

P

Pflanzen

Pflege

Wissensdatenbank

Einstellungen

2

Filter

Pflanzen

Wähle eine Pflanze

Standort

Wähle einen Standort

Aktion

Wähle eine Aktion

Anstehende Pflegeaufgaben

Pflanze	Standort	Aktion	Fälligkeitsdatum	Aktionen
Fensterblatt	Esszimmer	Gießen	25.10.2024	Erledigen
Fensterblatt	Esszimmer	Düngen	01.11.2024	Erledigen

Historie

Pflanze	Aktion	Erledigt am	Notizen
Fensterblatt	Standort geändert	18.10.2024	Notiz anzeigen

P

Pflanzen

Pflege

Wissensdatenbank

Einstellungen

2

Filter

Pflanzen

Wähle eine Pflanze

Standort

Wähle einen Standort

Aktion

Wähle eine Aktion

Anstehende Pflegeaufgaben

Pflanze	Standort	Aktion	Fälligkeitsdatum	Aktionen
Fensterblatt	Esszimmer	Düngen	01.11.2024	Erledigen

Historie

Pflanze	Aktion	Erledigt am	Notizen
Fensterblatt	Gießen	20.10.2024	Notiz hinzufügen
Fensterblatt	Standort geändert	18.10.2024	Notiz anzeigen

2.3.3 Notizen verwalten

Der Button „Notiz hinzufügen“ ermöglicht es Ihnen, zusätzliche Informationen zu einer Pflegeaufgabe zu hinterlegen. Wählen Sie eine Aufgabe aus der Tabelle der anstehenden Pflegeaufgaben aus und klicken Sie auf „Notiz hinzufügen“. Sie können dann eine Textnotiz eingeben, die relevante Details zur Aufgabe enthält, beispielsweise besondere Beobachtungen oder Hinweise zur Pflanze. Diese Notiz wird gespeichert und kann jederzeit in der Historie der erledigten Aufgaben eingesehen werden. So können Sie wichtige Details zur Pflanzenpflege dokumentieren und später nachvollziehen.

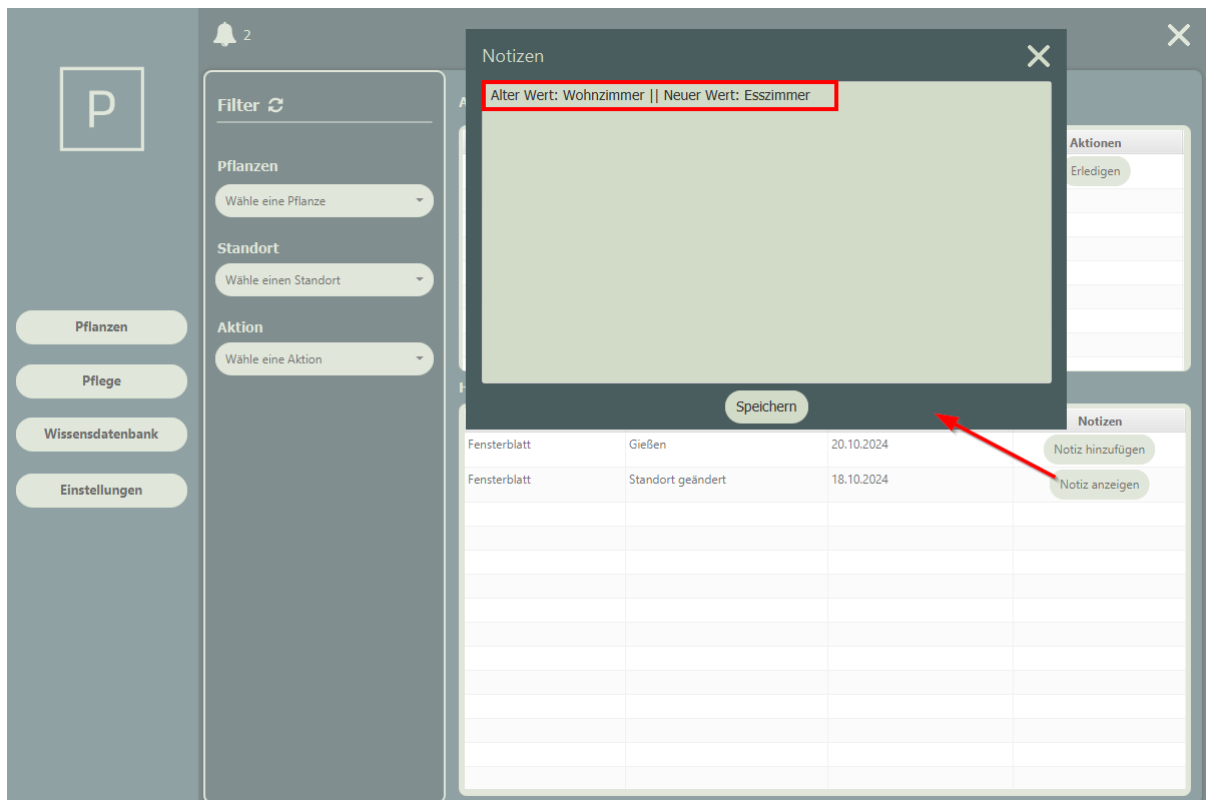
The screenshot shows the 'Pflanzen' application interface. On the left is a sidebar with a 'P' logo and buttons for 'Pflanzen', 'Pflege', 'Wissensdatenbank', and 'Einstellungen'. The main area has a 'Filter' section with dropdowns for 'Pflanzen', 'Standort', and 'Aktion'. Below this are two tables. The first table, 'Anstehende Pflegeaufgaben', has columns for 'Pflanze', 'Standort', 'Aktion', 'Fälligkeitsdatum', and 'Aktionen'. It contains one row for 'Fensterblatt' in the 'Esszimmer' to be 'Düngt' on '01.11.2024', with an 'Erledigen' button. The second table, 'Historie', has columns for 'Pflanze', 'Aktion', 'Erledigt am', and 'Notizen'. It contains two rows: one for 'Fensterblatt' where 'Gießen' was completed on '20.10.2024' (highlighted in yellow), and another for 'Fensterblatt' where 'Standort geändert' was completed on '18.10.2024'. Both history rows have buttons to 'Notiz hinzufügen' or 'Notiz anzeigen'.

Pflanze	Standort	Aktion	Fälligkeitsdatum	Aktionen
Fensterblatt	Esszimmer	Düngt	01.11.2024	Erledigen

Pflanze	Aktion	Erledigt am	Notizen
Fensterblatt	Gießen	20.10.2024	Notiz hinzufügen
Fensterblatt	Standort geändert	18.10.2024	Notiz anzeigen

Falls bereits eine Notiz zu einer Pflegeaufgabe vorhanden ist, ändert sich der Button-Text zu „Notiz anzeigen“. Hier können Sie die bereits hinterlegte Notiz einsehen, bearbeiten und bei Bedarf aktualisieren. Dies ermöglicht es Ihnen, bestehende Informationen zu ändern oder neue Details hinzuzufügen.

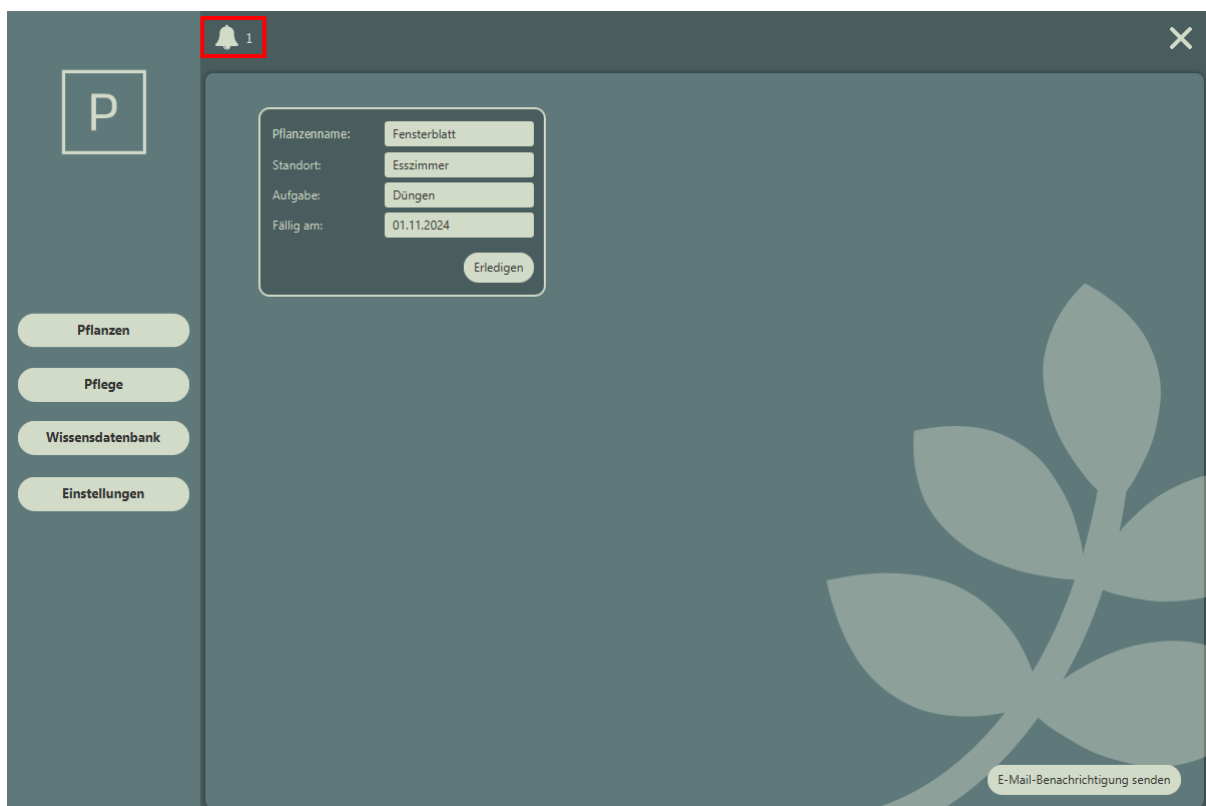
Änderungen im Pflanzenprofil (Name, Standort, Gieß- oder Düngintervalle) werden automatisch in einer Notiz festgehalten. So können Sie nachvollziehen, wann und welche Anpassungen an Ihren Pflanzen vorgenommen wurden.



2.3.4 Pflegebenachrichtigungen

In PlantPal gibt es zwei Möglichkeiten, wie Sie Benachrichtigungen zu anstehenden Aufgaben und wichtigen Ereignissen erhalten können: direkt in der Anwendung oder per E-Mail. Beide Benachrichtigungsarten helfen Ihnen dabei, keine wichtige Pflegemaßnahme für Ihre Pflanzen zu verpassen.

Wenn Sie die In-App-Benachrichtigung aktiviert haben, erscheint in der Anwendung ein Benachrichtigungs-Badge. Dieses Badge zeigt die Anzahl der anstehenden Pflegeaufgaben an. Sie erhalten so direkt beim Öffnen der Anwendung eine Übersicht über alle Aufgaben, die in den nächsten Tagen anstehen. Die Anzahl der Tage im Voraus, zu denen eine Erinnerung angezeigt werden soll, können Sie ebenfalls in den Einstellungen anpassen (Kapitel 2.5)



Darüber hinaus gibt es die Möglichkeit, Benachrichtigungen per E-Mail zu erhalten. Diese Option ist praktisch, wenn Sie auch außerhalb der Anwendung informiert werden möchten. Sie können festlegen, an welche E-Mail-Adresse die Benachrichtigungen gesendet werden sollen und von welcher Absenderadresse diese versendet werden. E-Mail-Benachrichtigungen können nur manuell über den Button „E-Mail-Benachrichtigung senden“ in der Anwendung verschickt werden, falls Sie sicherstellen möchten, dass keine wichtige Aufgabe vergessen wird.

P

Pflanzen

Pflege

Wissensdatenbank

Einstellungen

1

Pflanzenname:

Fensterblatt

Standort:

Esszimmer

Aufgabe:

Düngen

Fällig am:

01.11.2024

Erledigen

E-Mail-Benachrichtigung senden

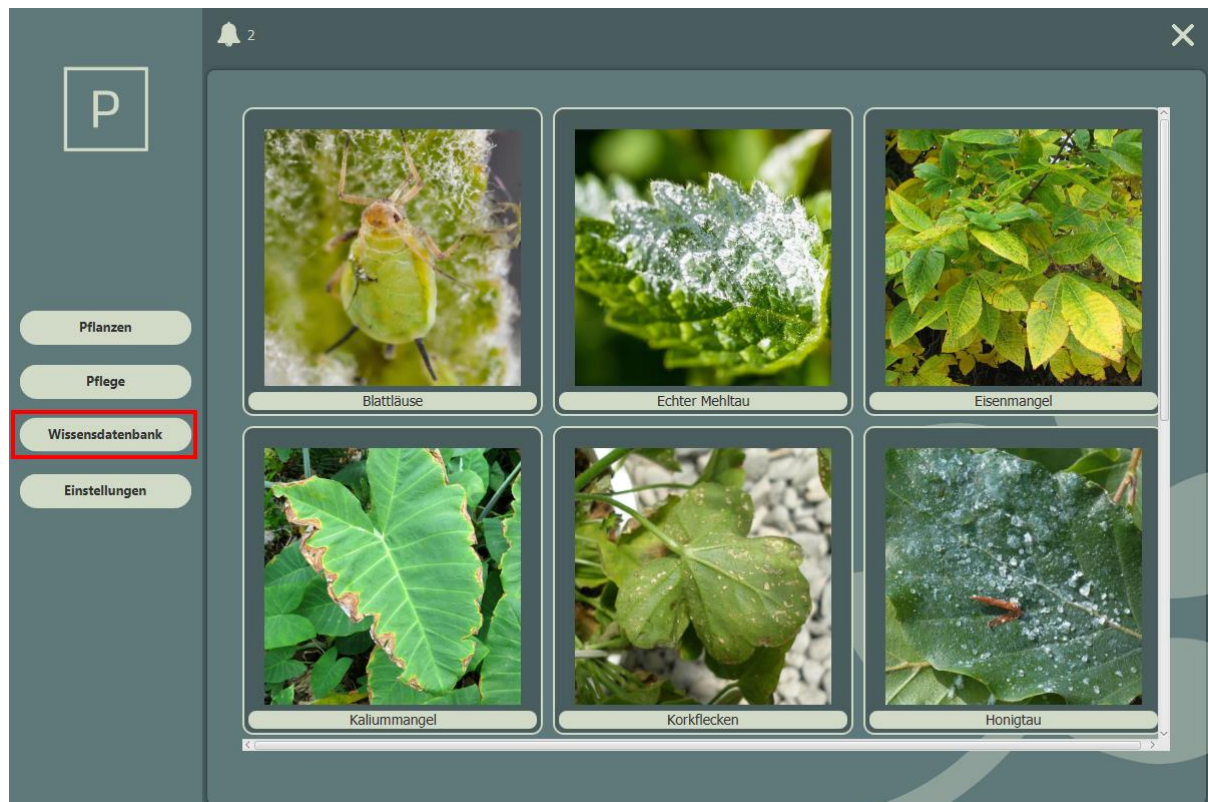
28

2.4. Sicht „Wissensdatenbank“

Die Wissensdatenbank in PlantPal bietet Ihnen Informationen zur Pflege Ihrer Pflanzen sowie zur Bekämpfung der häufigsten Schädlinge und Krankheiten. Sie hilft Ihnen, Symptome zu erkennen und passende Behandlungsmethoden zu finden, sodass Sie Ihre Pflanzen optimal versorgen können.

Die Wissensdatenbank ist in Kacheln organisiert, die die verschiedenen Krankheiten und Schädlingsbilder darstellen. Jede Kachel zeigt den Namen und ein Bild des jeweiligen Themas. Ein Klick auf eine Kachel öffnet eine Detailansicht mit umfassenden Informationen, die Ihnen bei der Behandlung Ihrer Pflanzen helfen.

In der Detailansicht erhalten Sie detaillierte Beschreibungen zu Symptomen, die bei der Diagnose helfen, sowie konkrete Hinweise zur Behandlung. Darüber hinaus gibt es Links zur Quelle für weitere Informationen, falls Sie tiefer in ein Thema einsteigen möchten.



P


2

Pflanzen


Pflege

Wissensdatenbank


Einstellungen



Blattläuse




Korkflecken



Honigtau

Kaliummangel



Details: Blattläuse

Symptome / Erkennung:

Blattläuse sind winzige, grüne, schwarze oder gelbe Insekten, die sich auf den Blättern und Trieben ansiedeln. Sie hinterlassen einen klebrigen Honigtau, der zu Rußpilzen führen kann. Befallene Pflanzen zeigen oft verkrüppelte oder gelbe Blätter.

Behandlung:

- Natürliche Feinde wie Marienkäfer einsetzen.
- Betroffene Pflanzen mit einer Seifenlösung besprühen.
- Bei starkem Befall spezielle Insektizide verwenden.

Quelle / Weitere Infos:

<https://www.pflanzen-koelle.de/ratgeber/pflanzendoktor/schaedlinge-u...>

2.5. Sicht „Einstellungen“

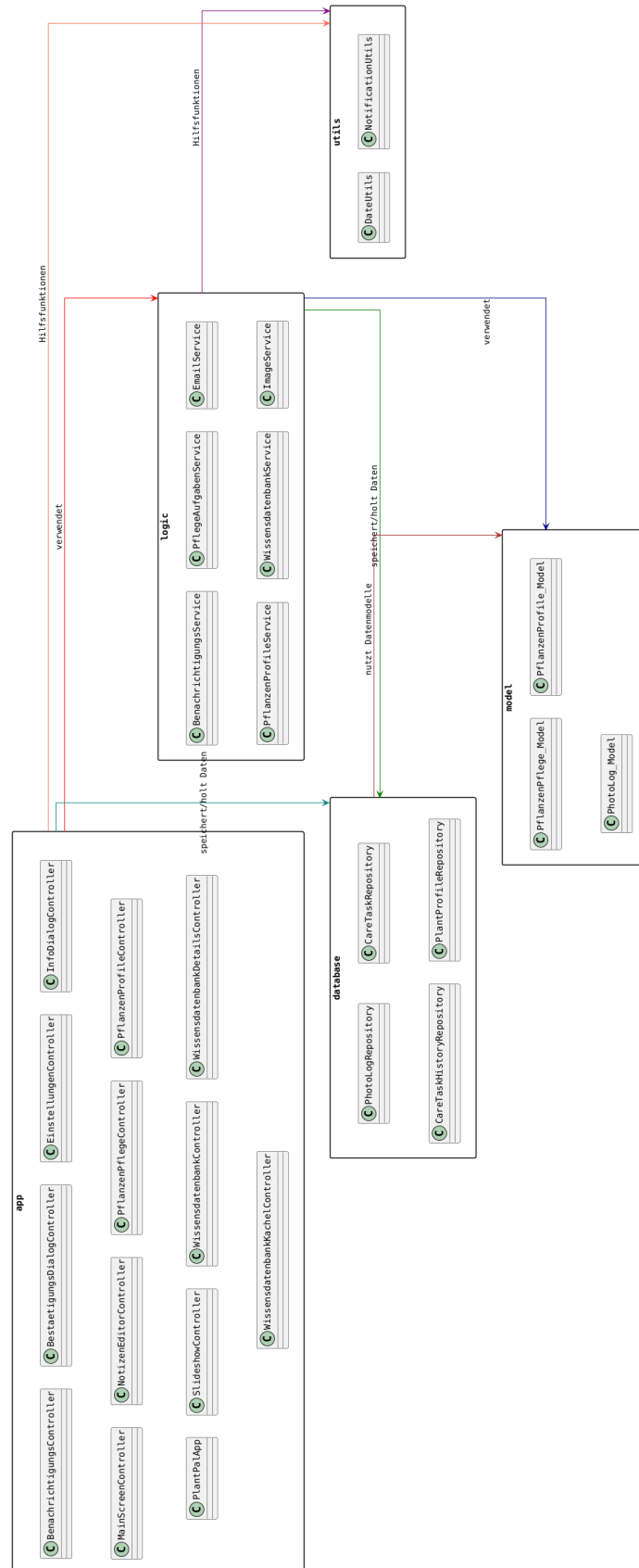
In den Einstellungen von PlantPal können Sie Ihre Präferenzen zur Anwendung individuell anpassen. Sie haben hier die Möglichkeit, wichtige Benachrichtigungs- und E-Mail-Einstellungen festzulegen und Ihr Benutzerprofil zu konfigurieren.

1	In-App-Benachrichtigungen	Checkbox zur Aktivierung von Benachrichtigungen direkt in der Anwendung. Wenn diese aktiviert ist, erhalten Sie Hinweise zu fälligen Aufgaben oder Änderungen innerhalb von PlantPal im Benachrichtigungs-Badge
2	Tage vor Erinnerung	Auswahlbox, die angezeigt wird, wenn die In-App-Benachrichtigung aktiviert ist. Hier können Sie festlegen, wie viele Tage im Voraus Sie eine Erinnerung erhalten möchten.
3	Mail-Benachrichtigung	Checkbox zur Aktivierung von E-Mail-Benachrichtigungen. Diese Funktion ist nur aktivierbar, wenn auch die In-App-Benachrichtigungen aktiviert sind. Sie erhalten Benachrichtigungen per E-Mail über anstehende Aufgaben.
4	Empfänger	E-Mail-Adresse, an die die Mail-Benachrichtigungen gesendet werden sollen.

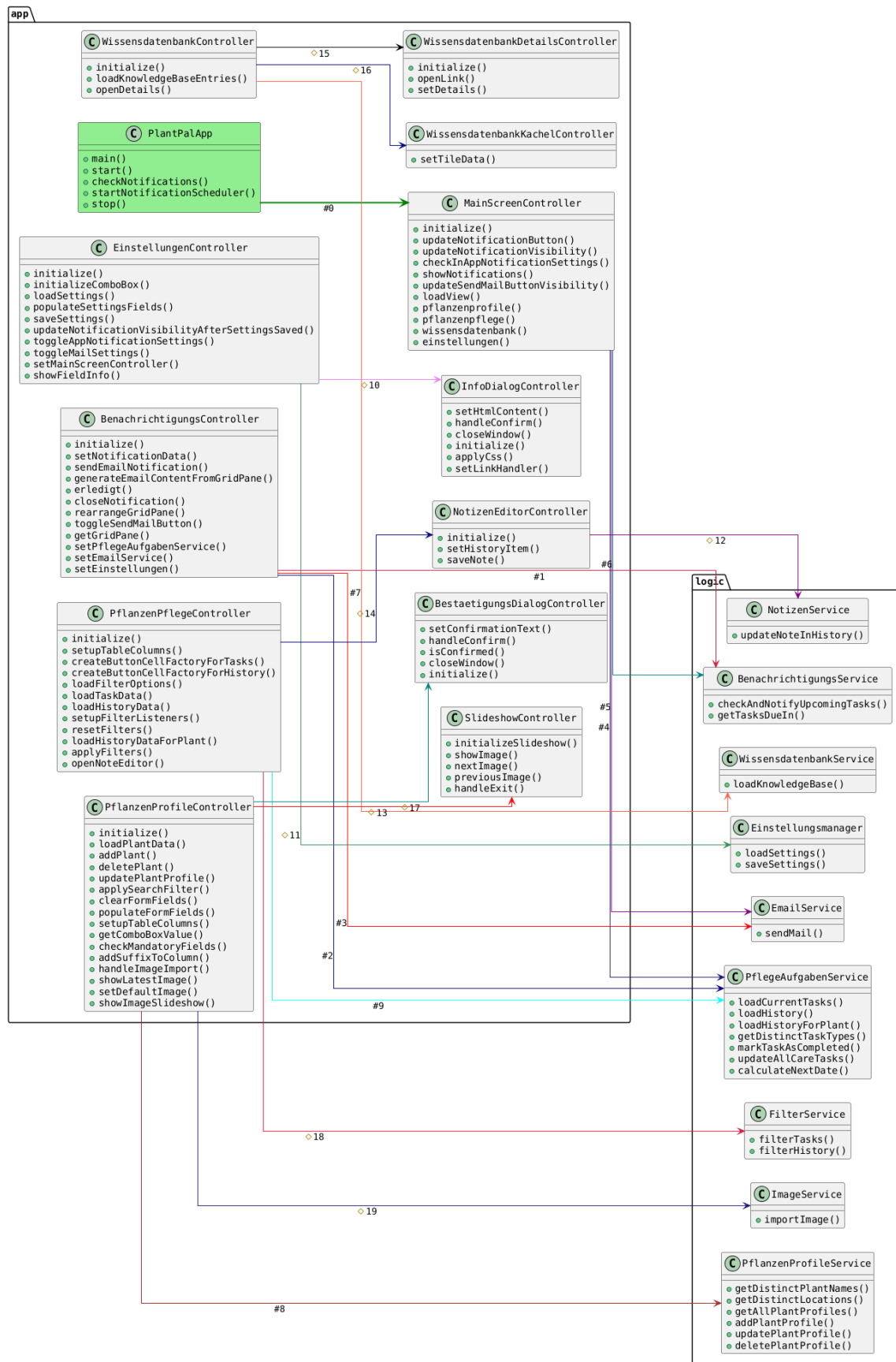
5	Absender	E-Mail-Adresse, von der die Mail-Benachrichtigungen versendet werden.
6	Info	Bietet zusätzliche Informationen zur Mailjet-Konfiguration. Bei Klick öffnet sich ein Fenster, das die Bedeutung der Felder erläutert und Hinweise zur Nutzung gibt.
7	API-Schlüssel	Passwortfeld zur Eingabe des öffentlichen API-Schlüssels von Mailjet, um den E-Mail-Dienst zu konfigurieren.
8	Privater API-Schlüssel	Passwortfeld zur Eingabe des privaten API-Schlüssels, der zur Authentifizierung bei Mailjet benötigt wird.
9	Speichern	Speichert alle vorgenommenen Änderungen. Sollte eine erforderliche Eingabe fehlen, wenn die Mail-Benachrichtigung aktiviert ist, wird eine entsprechende Fehlermeldung angezeigt.

Anhang

Anhang 1: UML-Klassendiagramm der abstrakten Systemarchitektur

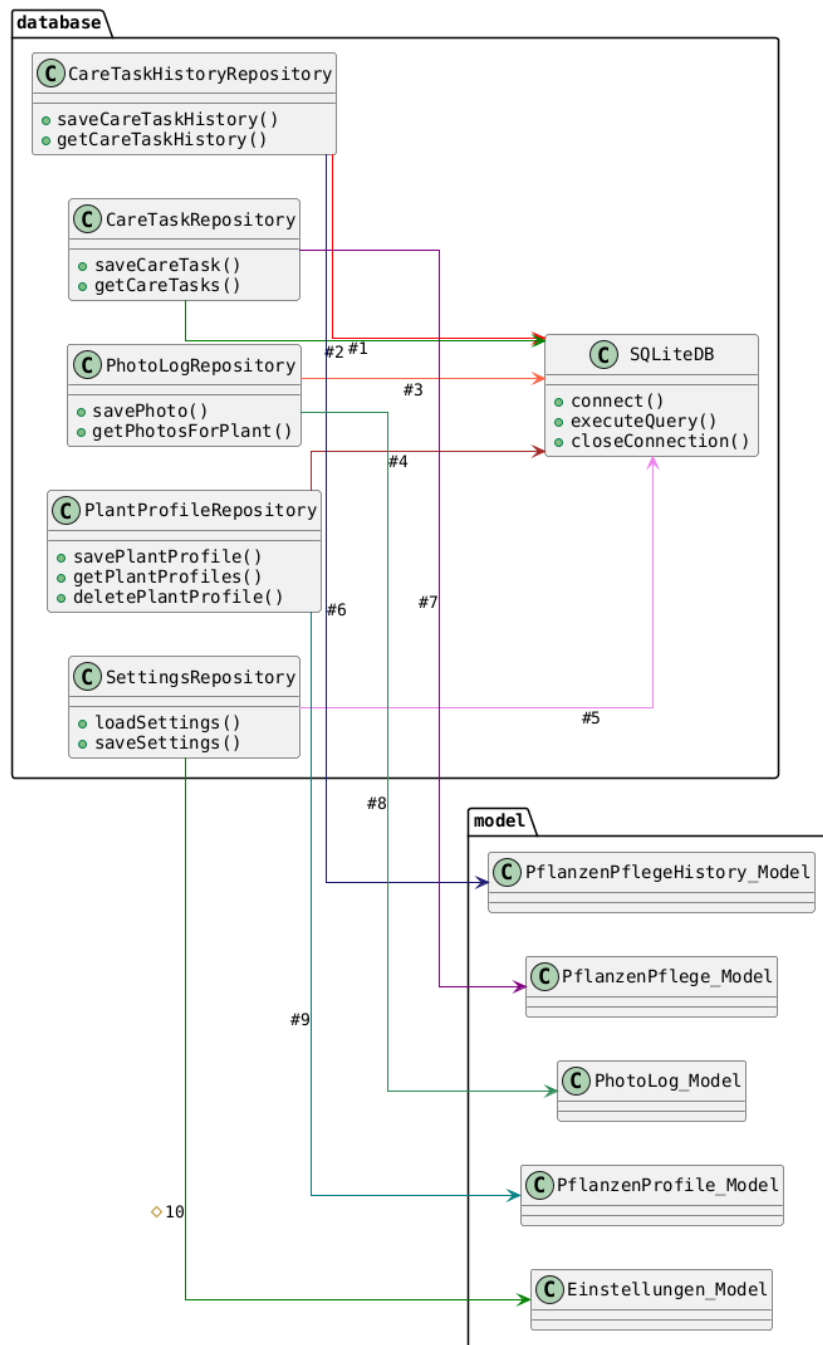


Anhang 2: UML-Klassendiagramm der Steuerungs- und Geschäftslogikmodule



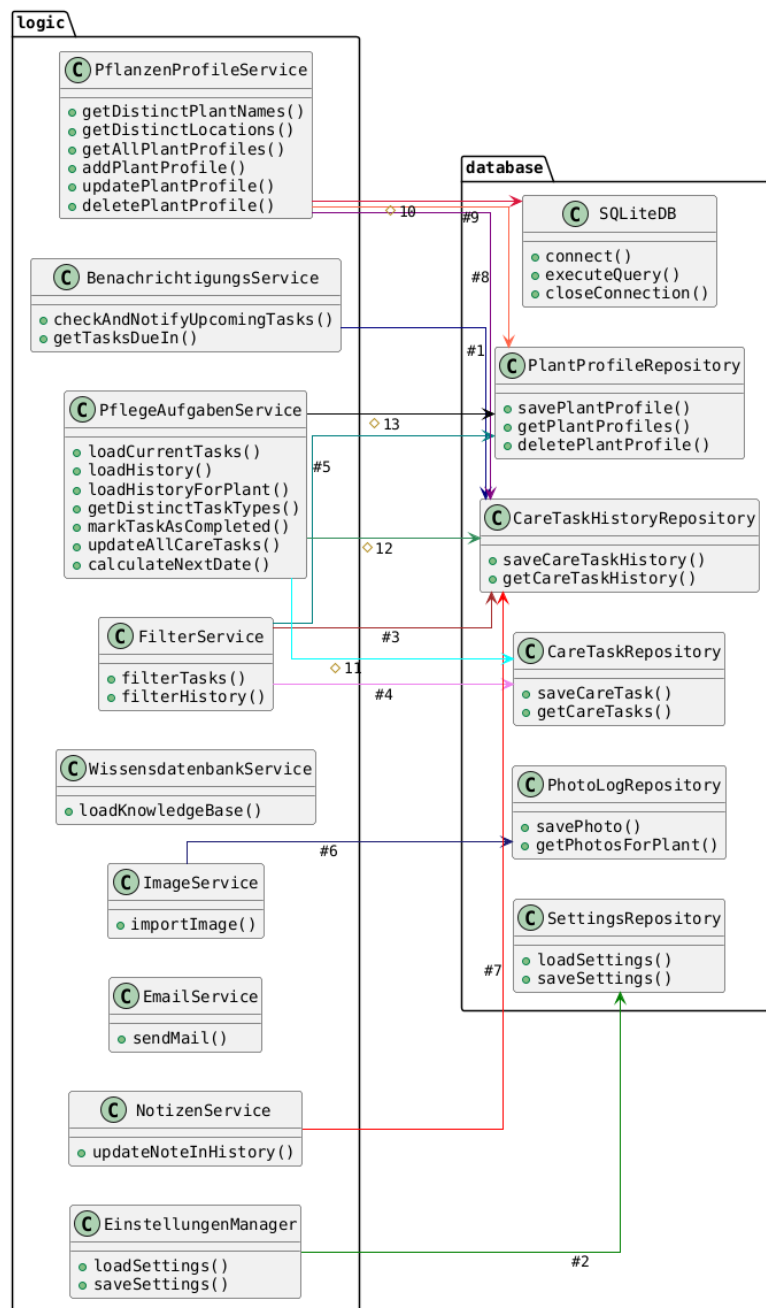
- #0 Die Anwendung startet und verbindet sich mit dem MainScreenController.
- #1 Der Benachrichtigungscontroller verwendet den Benachrichtigungsservice für das Senden von Benachrichtigungen.
- #2 Der Benachrichtigungscontroller interagiert mit dem Pflegeaufgabenservice.
- #3 Der Benachrichtigungscontroller sendet E-Mails über den EmailService.
- #4 Der MainScreenController verwendet den Pflegeaufgabenservice zur Anzeige und Bearbeitung von Pflegeaufgaben.
- #5 Der MainScreenController interagiert mit dem EmailService, um Benachrichtigungen per E-Mail zu versenden.
- #6 Der MainScreenController verwendet den Benachrichtigungsservice für die Verwaltung von Benachrichtigungen.
- #7 Der Wissensdatenbankcontroller interagiert mit dem Wissensdatenbankservice, um Daten aus der Wissensdatenbank zu laden.
- #8 Der Pflanzenprofilcontroller verwendet den Pflanzenprofilservice zur Verwaltung von Pflanzenprofilen.
- #9 Der Pflanzenpflegecontroller interagiert mit dem Pflegeaufgabenservice zur Verwaltung von Pflegeaufgaben.
- #10 Der EinstellungenController interagiert mit dem InfoDialogController zur Erzeugung vom Informationsdialog für die Maleinstellungen-Info.
- #11 Der Einstellungscontroller verwendet den Einstellungsmanager, um Benutzereinstellungen zu laden und zu speichern
- #12 Der Notizeneditorcontroller verwendet den Notizenservice zur Verwaltung von Notizen.
- #13 Der PflanzenPflegeController verwendet den SlideshowController um die Bilderslideshow für eine Pflanze zu öffnen
- #14 Der PflanzenPflegeController verwendet den NotizenEditorController um Notizen in der Historie zu verwalten.
- #15 Der WissensdatenbankController verwendet den WissensdatenbankDetailsController um die Detailansicht der Wissensdatenbank zu öffnen
- #16 Der WissensdatenbankController verwendet den WissensdatenbankKachelController um die Einträge in der DB zu laden und im GridPane der Wissensdatenbank anzuzeigen
- #17 Der Pflanzenprofilcontroller verwendet der BestaetigungsDialogController um den Bestätigungsdialog beim Löschen eines Pflanzenprofils anzuzeigen
- #18 Der PflanzenPflegeController verwendet den FilterService um die Aufgaben in der Pflanzenpflege zu filtern.
- #19 Der PflanzenProfileController verwendet den ImageService um Bilder hochzuladen.

Anhang 3: UML-Klassendiagramm der Datenbank- und Modelmodule



- #1 Interagiert mit der SQLite-Datenbank, um Daten zu Schreiben oder zu Lesen
- #2 Interagiert mit der SQLite-Datenbank, um Daten zu Schreiben oder zu Lesen
- #3 Interagiert mit der SQLite-Datenbank, um Daten zu Schreiben oder zu Lesen
- #4 Interagiert mit der SQLite-Datenbank, um Daten zu Schreiben oder zu Lesen
- #5 Interagiert mit der SQLite-Datenbank, um Daten zu Schreiben oder zu Lesen
- #6 Interagiert mit PflanzenPflegeHistory_Model zum Lesen von Historieneinträgen
- #7 Interagiert mit PflanzenPflege_Model zum Lesen und Filtern von Pflegeaufgaben
- #8 Interagiert mit PhotoLog_Model zum Lesen der Bildpfades aus der DB
- #9 Interagiert mit PflanzenProfile_Model zum Lesen der Pflanzenprofilen
- #10 Interagiert mit Einstellungen_Model zum Lesen und Aktualisieren von Einstellungsdaten

Anhang 4: UML-Klassendiagramm der Geschäftslogik- und Datenbankmodule



- #1 Holt Historie der Pflegeaufgaben für Benachrichtigungen
- #2 Lädt oder speichert Benutzereinstellungen
- #3 Filtert Aufgabenhistorie
- #4 Filtert aktuelle Pflegeaufgaben
- #5 Filtert Pflanzenprofile
- #6 Importiert und speichert Fotos der Pflanzen
- #7 Aktualisiert Notizen in der Aufgabenhistorie
- #8 Interagiert mit Aufgabenhistorie
- #9 Fügt Pflanzprofile hinzu, aktualisiert oder löscht sie
- #10 Stellt eine Verbindung zur Datenbank her
- #11 Verwalten von Pflegeaufgaben
- #12 Historie der Pflegeaufgaben verwalten
- #13 Verknüpft Aufgaben mit Pflanzenprofilen

Anhang 5: UML-Klassendiagramm Gesamtdarstellung

