Technischer Bericht II

GardenPlanner

Elias Csomor
Stellvertretender Projektleiter / Entwickler
csomoeli@students.zhaw.ch

Philippe Giavarini
Projektleiter / Entwickler
giavaphi@students.zhaw.ch

David Guler
Entwickler
gulerdav@students.zhaw.ch

Gian-Andrea Hutter
Entwickler
huttegia@students.zhaw.ch

Roman Schenk
Entwickler
schrom01@students.zhaw.ch

ZHAW – School of Engineering
Gruppe: FivePlants
Dozent: Ferri Reto (feit)

Klasse: IT21b_WIN

Abgabe: 12 Dezember 2022

Lehrgang: Projektmodul 3 (HS22)

Abstract

In the wake of the recent supply chain disruptions due to the worsening effects of climate change compounded by the pandemic, interest in efficient home gardening has increased among the general population. This project aims to help both new and experienced gardeners increase the productivity and yield of their gardens by creating a garden planning software which automatically schedules the tasks required from sowing to harvest. While similar applications exist, this project additionally adjusts the schedule based on weather forecasts, ensuring minimal wasted resources and crop loss. The software was designed with the primary goal of ease of use; the user should be guided to creating a valid garden plan by the graphical interface. This was achieved by initially writing a series of use cases detailing the steps by which a user would navigate the interface. The beta release of the software, which is the subject of this report, lays the foundation for further development. The focus will be on adding a visualisation of the user's garden and giving helpful feedback for invalid inputs.

Inhaltsverzeichnis

1 Projektidee

1.1 Ausgangslage

Ständig sind wir mit der Thematik des Klimawandels konfrontiert und überall wird Nachhaltigkeit grossgeschrieben, so auch bei der Ernährung. Heutzutage wollen viele Menschen wissen, wie die Produkte hergestellt werden und was alles darin enthalten ist.

Daher versuchen viele Personen Gemüse und Obst selbst anzupflanzen. Allerdings ist eine gute Planung erforderlich, damit nicht alles Gemüse gleichzeitig erntereif ist. Die Gewächse brauchen ausreichend Wasser, müssen gedüngt und vor möglichen Unwettern geschützt werden. Es gibt bereits einzelne Apps, die Tipps geben, wie das Gemüse angebaut wird und womit sie gedüngt werden. Dazu kommen Apps, die zeigen, welche Pflanzen am besten in welchen Klimazonen gedeihen.

1.2 Idee

Ziel dieses Projekts ist es, eine App zu entwickeln, mit welcher der Gemüseanbau selbstständig geplant werden kann. Mit dieser Anwendung können Kund:innen ihre Gartenfläche optimal planen und direkt bestimmen, was als nächstes angepflanzt werden soll. Weiter soll sie die Nutzenden laufend über die Wachstumsphase informieren und Tipps geben, was in der aktuellen Phase berücksichtigt werden soll. Wird ein Schädlingsbefall entdeckt, empfiehlt die App Massnahmen zur Bekämpfung des Schädlings. Im Vorfeld können die Anwender:innen sich in der App informieren, was sie vorbeugend machen können. Das schont Ressourcen und der Boden wird weniger mit Pestiziden belastet.

Eine Einbindung eines Wetterdienstes ermöglicht es die Bewässerung genau zu planen und auch hier Wasser zu sparen, damit die Pflanzen in einer niederschlagsreichen Zeit nicht zusätzlich gegossen werden. Wenn die Pflanzen erntereif sind, sollen die Benutzer:innen informiert werden.

So kann genauer geplant werden, wann die Nutzenden welches Gemüse vorzugsweise ernten möchten und wann sie es dafür anpflanzen müssen. Die Applikation verschafft einen einfachen Überblick über alle Pflanzen und in welchem Stadium diese sind. Dadurch verpassen die Kund:innen die Erntephase nicht und haben nicht in einem Monat alles Gemüse und im nächsten Monat gar nichts.

In einer Community können sich die Kund:innen über Tipps zur Anpflanzung verschiedener Pflanzen, zur Schädlingsbekämpfung oder der Anordnung der Gewächse im Beet austauschen.

1.3 Kundennutzen

In diesem Abschnitt werden die Vorteile aufgelistet, die die Benutzer:innen haben, wenn sie die Applikation verwenden.

- Das Planen der Anpflanzung wird einfacher.
- Die Applikation erinnert daran, wann welche Tätigkeiten im Garten anfallen.
- Der zeitliche Aufwand wird minimiert, da die Kund:innen nicht täglich zu ihrem Garten schauen müssen.

- Die Nährstoffe im Boden können optimal genutzt werden, da die Applikation vorschlägt, welche Pflanzen als nächstes gepflanzt werden sollen.
- Bei der Bewässerung kann Wasser gespart werden, da die Bewässerungsaufgaben nach dem Regen aufgeschoben werden.
- Mit gezielten Massnahmen gegen Schädlinge, wird der Pestizideinsatz minimiert und so die Umwelt weniger belastet.

1.4 Stand der Technik / Konkurrenzanalyse

In Tabelle 1 sind die grössten Konkurrenten und ihre Dienstleistungen aufgelistet. In der Applikation, welche Gegenstand dieses Projekts ist, werden Dienstleistungen der bestehenden Anwendungen zusammengefasst und durch Einbindung eines Wetterdiensts erweitert.

Tabelle 1: Dienstleistungsübersicht der Konkurrenzprodukte

Name	Dienstleistungen
GrowVeg Garden Planner **[1]**	- Pflanzen im Garten anlegen - Aufgabenliste - Visuelle Planung des Gartens inklusive Gartenhäuser, Pfade, und Bewässerungssysteme - Die Wachstumsinformationen der Pflanzen an lokales Klima, Gartenhäuser angepasst - Erinnerungen per E-Mail
VegPlotter **[2]**	- Browseranwendung - Beete frei formbar - Monatliche Planung
Smartgardener **[3]**	- Browseranwendung - Wachstum der Pflanzen nicht an lokales Klima angepasst
Veggie Garden Planner **[4]**	- Nur für Mobilgeräte - Informationen zur Verträglichkeit mit anderen Pflanzen - Passt sich nicht ans lokale Klima an
Gardenverwaltung	- **Pflanzen anlegen** - **Wachstumsinformationen an lokales Klima angepasst** - **Aufgabenliste** - **Aufgaben bearbeitbar** - **Erinnerungen per E-Mail** - **Wettereinbindung**

1.5 Kontextszenario Gärtnerin (Laie)

Persona:

Lea, 21, Informatikstudentin, wohnt bei den Eltern in Benken. Ihre Eltern finden, sie solle mehr in der Natur machen, statt nur vor dem Bildschirm zu sitzen. Sie entschliesst sich, ihr eigenes Gemüse anzubauen. Als Einstieg möchte sie Zwiebeln anbauen, da diese in vielen ihrer Lieblingsgerichte Verwendung finden.

Lea hat auf ihrem Laptop die Anwendung “GardenPlanner” installiert, da sie ihr von einem Kollegen empfohlen wurde. In ihrem Garten hat sie eine Beet-Fläche, auf der sie gerne ihre Pflanzen anbauen möchte.

Lea startet die Anwendung und wird nach den Dimensionen ihrer Anbaufläche sowie dem Standort ihres Gartens gefragt. Weil sie möglichst rasch anpflanzen will, wechselt sie direkt in die Auswahl der

Pflanzen. Sie wählt die aktuelle Jahreszeit (Sommer) und entscheidet sich für die Zwiebeln.
Als Anpflanzdatum wählt sie das nächste Wochenende, weil sie vorher noch die Knollen besorgen muss.

Für die verschiedenen Tätigkeiten, die bis zur Ernte zu erledigen sind, erstellt ihr der “GardenPlanner” nun eine Liste. Zu diesen Tätigkeiten gehören vor allem das Bewässern und Düngen der Pflanzen.

Einmal pro Woche loggt sie sich nun in den “GardenPlanner” ein, markiert die durchgeführten Schritte, als erledigt und liest den Stand der Aufgaben für die nächste Woche. Einmal geschieht es, dass der “GardenPlanner” die Tätigkeiten angepasst hat: für nächste Woche hat der Wetterdienst starke Gewitter über der Region angekündigt, das Giessen muss deswegen nicht stattfinden. Dafür wird ihr dazu geraten, ihre Pflanzen mit einem Hagelschutz-Netz abzudecken, um sie vor allfälligen Hagelschäden zu schützen. Die anderen Wochen verlaufen nach ursprünglichem Plan.

Am Datum, an dem die Ernte eingetragen ist, geht sie in den Garten und erntet die ersten schon ganz reifen Gemüse. Als letzte Aktion bestätigt sie im “GardenPlanner” die Ernte.

1.6 Wirtschaftlichkeit

Hier werden die anfallenden Kosten und der mögliche Gewinn basierend auf den Einnahmen besprochen. Auch wird angegeben, nach wie viel Zeit mit einem Return on Investment gerechnet werden kann.

1.6.1 Kosten

Hier werden die anfallenden Kosten für das Projekt beschrieben (siehe Tabelle 2).

Der zeitliche Aufwand, für die Erstellung des Produktes, wird auf 1'800 Personenstunden geschätzt, ungefähr ein Jahr. Dabei werden 600 Stunden für die Beta-Version gerechnet und weitere 1200 Stunden, bis das fertige Produkt vermarktet werden kann. Bei einem Lohn von 100.- Franken pro Stunde, führt dies zu einmaligen Kosten von 180'000.- Franken.

Für die jährlich anfallenden Kosten werden die Wartung der Software und die Lizenzgebühr der Wetterdienst API berechnet.

Für die Kosten der Wetterdienst API referenzieren wir die Kosten eines potenziellen Anbieters: Meteonomiqs [5]. Bei geschätzten 1'500'000 monatlichen Anfragen an die API, werden mit Kosten von 450.- Franken pro Monat gerechnet.

Für Wartungsarbeiten werden 25 Stunden pro Monat geschätzt, was bei einem Lohn von 100.- Franken pro Stunde zu weiteren 2'500.- Franken an monatlichen Kosten führt. Unter Wartungsarbeiten wird verstanden, das Produkt auf eine neue Version zu bringen, Fehler zu beheben, und dass die Funktionsfähigkeit der Software gewährleistet bleibt.

Zusammenfassend, belaufen sich die einmaligen Entwicklungskosten auf 180’000.- Franken und die fortlaufenden Kosten auf 35’400.- Franken pro Jahr.

Tabelle 2: Anfallende Kosten

Einmalige Kosten		
Beschreibung (Stunden)	**Lohn (CHF)**	**Total (CHF)**
Entwicklung des Produktes (1'800)	100.-/Stunde	180'000.-
Total		**180'000.-**
Jährlich anfallende Kosten		
Beschreibung	**Kosten (CHF)/Monat**	**Total (CHF)/Jahr**
Lizenz für Wetterdienst API	450.-	5'400.-
Wartung (25 Stunden/Monat)	2’500.- (100.-/Stunde)	30'000.-
Total		**35'400.-**

1.6.2 Einnahmen

Das Produkt wird mit zwei Methoden gewinnbringend vermarktet (siehe Tabelle 3).

Mit der ersten Methode werden Verträge mit Werbetreibenden gemacht, welche ihre Werbungen im Produkt darstellen können. Das Produkt wird kostenlos angeboten und enthält nur die Grundfunktionalitäten. Es wird geplant, genügend Werbungen einzubinden, sodass ein Tausend-Kontakt-Preis (TKP) von 12.- Franken erzielt werden kann. Das bedeutet, dass 12.- Franken eingenommen werden, wenn das Produkt 1’000-mal aktiv verwendet wird. TKP bedeutet, dass pro 1'000 Aufrufe der Werbungen der bestimmte Betrag eingenommen wird. Der angegebene TKP setzt sich aus den Einnahmen von mehreren Werbetreibenden zusammen.

Mit der Schätzung, dass die Applikation täglich von durchschnittlich 6'000 aktiven Benutzer:innen mindestens einmal geöffnet wird, werden pro Tag (6 x 12.- Franken) 72.- Franken eingenommen. Welches zu Einnahmen von 2'160.- Franken pro Monat führt.

Die andere Vermarktungsmethode ist ein Abonnement, welches der Benutzer oder die Benutzerin pro Monat zahlt. Diese Version des Produkts wird werbefrei sein und den vollen Funktionsumfang enthalten. Mit Abonnementkosten von 4.- Franken pro Monat und einer Schätzung von mindestens 1'500 Abonnent:innen, werden monatliche Einnahmen von 6'000.- Franken erwartet.

Damit erhält man jährlich 26'000.- Franken von der ersten Methode und 72'000.- Franken von der zweiten Methode.

Tabelle 3: Jährliche Einnahmen

Beschreibung (mind. Anzahl Benutzer/-innen)	Einnahmen (CHF) pro Monat	Einnahmen (CHF) pro Jahr
Verwendung der Gratisversion (6'000 pro Tag)	2'160.- (12.- Franken pro 1'000 Aufrufe)	26'000.-
Kaufen das Abonnement (1'500 pro Monat)	6'000.- (4.- Franken pro Abo)	72'000.-
Total		**98'000.-**

1.6.3 Return on Investment

Unter Berücksichtigung von den jährlichen Kosten und den jährlichen Einnahmen, ist der Return on Investment möglich, nach drei Jahren.

2 Analyse

Die Anforderungen der potenziellen Nutzer und Nutzerinnen sind in ein Use-Case-Modell und zusätzliche Anforderungen unterteilt und werden im folgenden Kapitel dargestellt und erläutert.

2.1 Use-Case-Modell

Im Use-Case-Diagramm sind alle Use-Cases abgebildet (siehe Abbildung 1: Use-Case Diagramm).

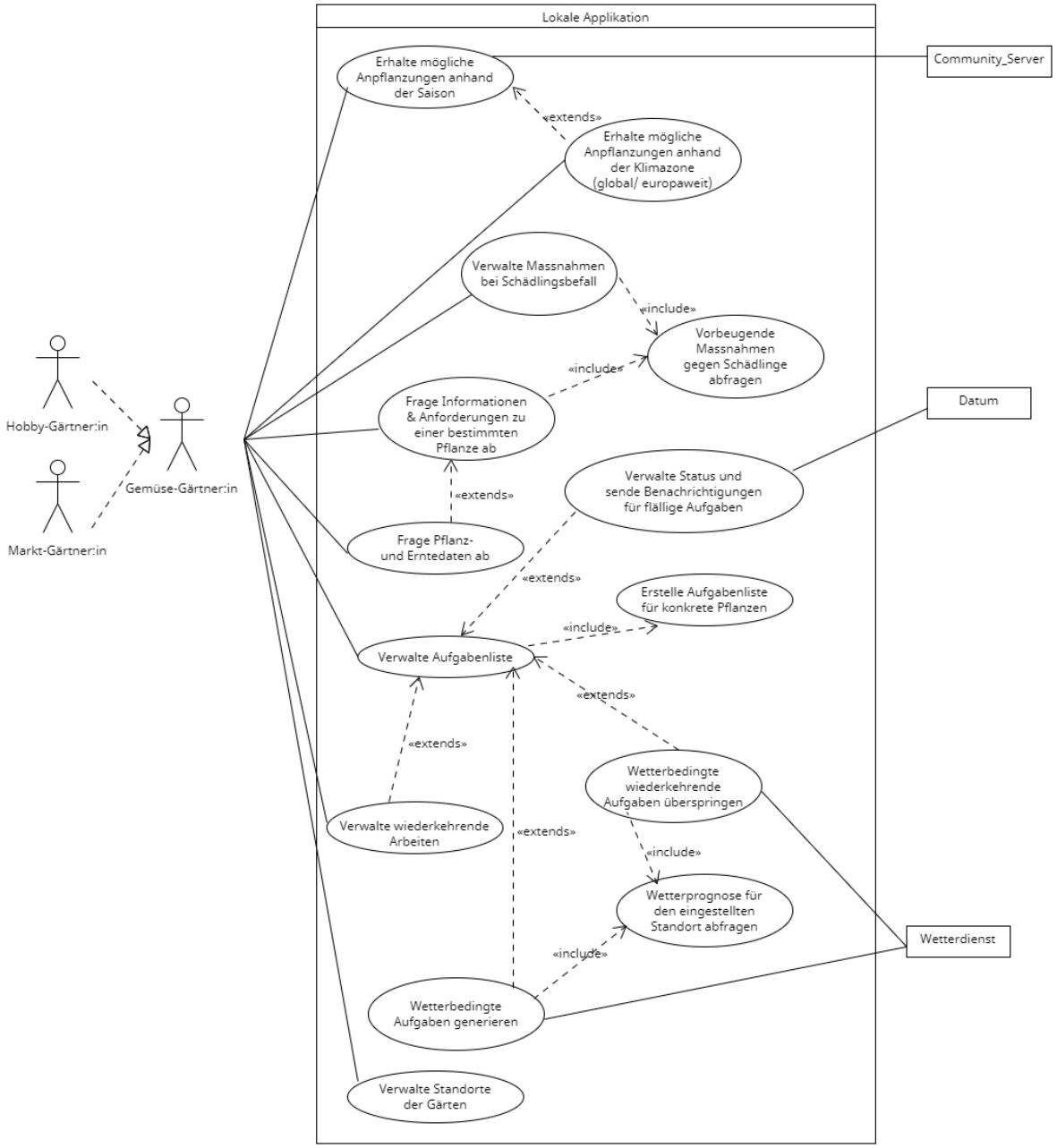

Abbildung 1: Use-Case Diagramm

Die im Diagramm dargestellten Use-Cases sind in vier Gruppen eingeteilt. Die Use-Cases sind zum Teil voneinander abhängig. Die Nummerierung ist so gewählt, dass die Use-Cases in dieser Reihenfolge umgesetzt werden können.

2.1.1 Use-Case-Gruppe 1

UC 1.0: Erhalte mögliche Anpflanzungen anhand der Saison

Der Gärtner oder die Gärtnerin öffnet die App und will neue Pflanzen anpflanzen. Die App zeigt die zur Verfügung stehenden Pflanzen an. Der Gärtner oder die Gärtnerin lässt entweder die Pflanzen in der App nach der aktuellen Saison sortieren oder sucht die gewünschte Pflanze anhand ihres Namens. Aus der sortierten Auswahl wählt der Gärtner oder die Gärtnerin aus, welche Pflanzen angebaut werden sollen. Diese werden in einer Liste für die geplanten Pflanzen gespeichert.

Sollten die Suchergebnisse nicht zufriedenstellend sein, kann zudem die Community Option aktiviert werden. Damit werden zusätzlich Einträge angezeigt, die von anderen Nutzern und Nutzerinnen erstellt und publiziert wurden. Diese Pflanzen können direkt in die «Liste der verfügbaren Pflanzen» importiert und bei Bedarf noch individualisiert werden.

Sollten auch die Community-Ergebnisse nicht zufriedenstellend sein, kann ein eigener Eintrag erstellt, abgespeichert und der Community zur Verfügung gestellt werden.

Dieser Use-Case ist im System Sequenzdiagramm (siehe Abbildung 2: System Sequenz Diagramm Use-Case 1.0) abgebildet.

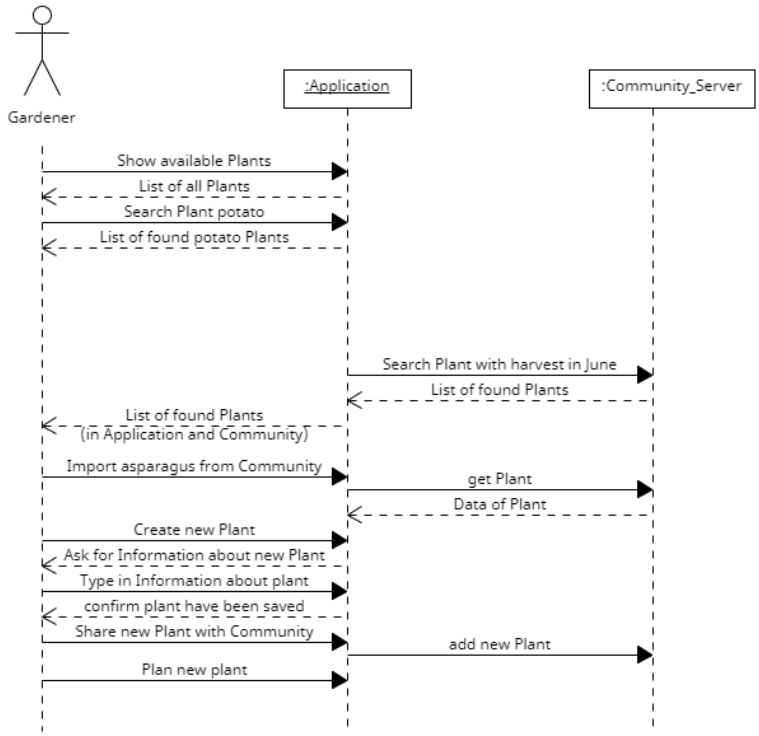

Abbildung 2: System Sequenz Diagramm Use-Case 1.0

UC 1.1: Erhalte mögliche Anpflanzungen anhand der Klimazone

Der Gärtner oder die Gärtnerin öffnet die App und möchte neue Pflanzen anpflanzen. Die App zeigt die zur Verfügung stehenden Pflanzen an. Der Gärtner oder die Gärtnerin gibt das Land und die Provinz ein, in der sich der Garten befindet und lässt die Pflanzen nach der Klimazone filtern. Aus der gefilterten Auswahl wählt er/sie aus, welche Pflanzen angebaut werden sollen. Diese werden in einer Liste für die geplanten Pflanzen gespeichert.

2.1.2 Use-Case-Gruppe 2

UC 2.0: Frage Informationen & Anforderungen zu einer bestimmten Pflanze ab
Der Gärtner oder die Gärtnerin tippt auf die Übersicht der ausgewählten Pflanzen. Die App zeigt alle ausgewählten Pflanzen an. Nach dem Anklicken einer Pflanze zeigt die Applikation alle Informationen zur Pflanze an (Anforderungen: Licht, Wasser, Erde; Informationen: Wachstumsphase, Pflanz und Erntezeit, mögliche Schädlinge).

UC 2.1: Plane Pflanz- und Erntedaten

Der Gärtner oder die Gärtnerin wählt eine Pflanze aus der Liste der möglichen Anpflanzungen aus und er/sie gibt an, ob das Anpflanzdatum oder das Erntedatum ausgewählt werden soll.

Wird das Anpflanzdatum eingegeben, zeigt die App vorgegebene Daten an. Nachdem ein Datum gewählt wurde, zeigt die App das Erntedatum an.

Wird das Erntedatum eingegeben, zeigt die App vorgegebene Daten an. Der Gärtner oder die Gärtnerin wählt das Datum aus. Die App zeigt das Datum an, an welchem die Pflanzen angepflanzt werden sollen, damit sie am gewünschten Datum geerntet werden kann.

Zusätzlich zum Datum gibt der Gärtner oder die Gärtnerin auch die Menge der Pflanzen an, die er/sie pflanzen möchte, damit dies auch später wieder abgerufen werden kann.

Dieser Use-Case ist im System Sequenzdiagramm (siehe Abbildung 3: System Sequenz Diagramm Use-Case 2.1) abgebildet.

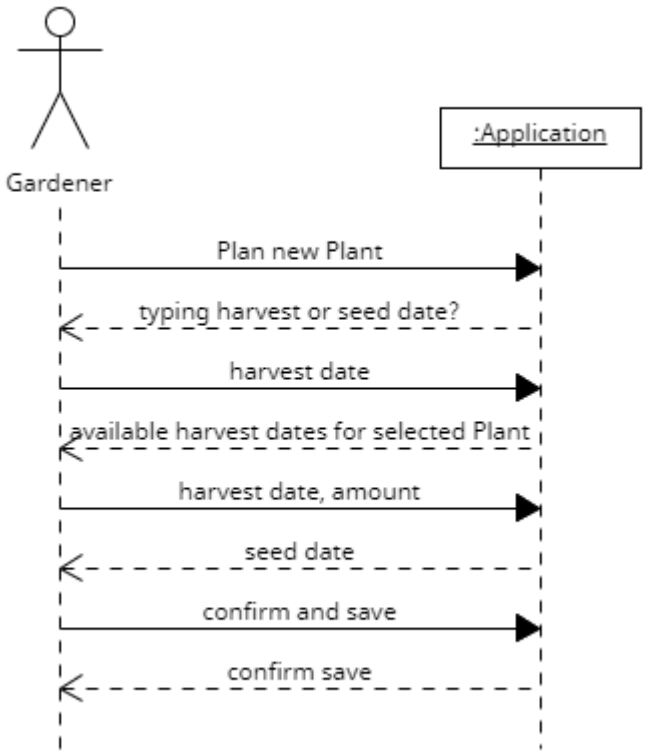

Abbildung 3: System Sequenz Diagramm Use-Case 2.1

2.1.3 Use-Case-Gruppe 3

UC 3.0: Erstelle Aufgabenliste für konkrete Pflanzen (fully dressed)

Umfang: Gartenverwaltung-Anwendung

Ebene: Anwenderziel

Primärakteur: Gärtner:in

Stakeholder und Interessen:

Gärtner:innen und andere Pflanzenkultivierungsinteressierte:

- Möchten das Pflanzen und Ernten einer Pflanze planen
- Möchten eine Liste der nötigen Schritte der Kultivierung der Pflanze erhalten

Vorbedingungen:

- Die Ernte- oder Anpflanzzeit muss wie im Use-Case 2.1 (Kapitel 2.1.2) beschrieben ausgewählt worden sein.
- Eine Pflanze muss anhand der Klimazone bestimmt worden sein.

Nachbedingungen:

- Eine Aufgabenliste anhand der gegebenen Angaben wird abgespeichert.
- Aufgaben der nächsten 7 Tage werden im Zeitplan angegeben.

Standardablauf:

1. Der Benutzer oder die Benutzerin bestimmt die Pflanze, Klimazone und Anpflanzzeit.
2. Die Applikation zeigt detaillierte Informationen der Pflanzen an.
3. Die Applikation erstellt eine Aufgabenliste, sortiert nach Wachstumsphasen.
4. Der Benutzer oder die Benutzerin navigiert auf den Zeitplaner.
5. Die Applikation zeigt alle Aufgaben (von allen Pflanzen) der nächsten Woche im Zeitplaner an.
6. Der Benutzer oder die Benutzerin wählt eine bestimmte Pflanze aus.
7. Die Applikation zeigt nur die Aufgaben an, die mit der ausgewählten Pflanze verknüpft sind.

Diese Schritte sind im System Sequenzdiagramm (siehe Abbildung 4: System Sequenzdiagramm Use-Case 3.0) abgebildet.

Erweiterungen:

- Der Benutzer oder die Benutzerin kann die bestehenden Aufgaben verweigern (abgedeckt in UC 3.1)
- Der Benutzer oder die Benutzerin kann das Intervall der bestehenden Aufgaben anpassen. (abgedeckt in UC 3.2)
- Der Benutzer oder die Benutzerin kann neue Aufgaben erstellen und löschen. (abgedeckt in UC 3.1)
- Wenn die Anwendung geschlossen wird und erneut geöffnet wird, ist die Aufgabenliste immer noch abrufbar.
- Wenn die Aufgabenliste am nächsten Tag abgerufen wird, soll der Zeitplan die richtigen korrespondierenden Aufgaben angeben.

Spezielle Anforderungen:

- Die Aufgabenliste muss alle geplanten Pflanzen beinhalten.
- Die Aufgabenliste kann anhand der Pflanze gefiltert werden.

Liste der Technik- und Datenvariationen:

- Die Aufgabenliste soll auf dem lokalen Gerät abgespeichert werden.

Häufigkeit des Auftretens: Einmalig, wenn die Kultivierung einer neuen Pflanze geplant wird.

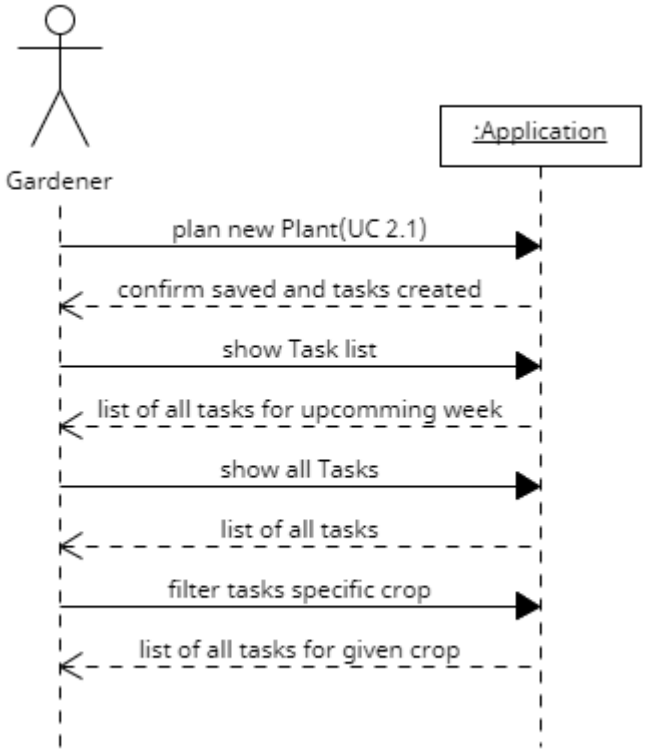

Abbildung 4: System Sequenzdiagramm Use-Case 3.0

UC 3.1: Verwalte Aufgabenliste
Der Benutzer oder die Benutzerin kann die volle Aufgabenliste anhand der detaillierten Ansicht der geplanten Pflanze begutachten. Dabei kann der Benutzer oder die Benutzerin entscheiden, welche der Aufgaben er/sie weiterhin ausführen möchte oder ob diese weggelassen werden sollen.

Der Benutzer oder die Benutzerin kann dabei auch neue Aufgaben in den verschiedenen Wachstumsphasen erstellen. Diese können nachträglich wieder bearbeitet werden.

UC 3.2: Verwalte wiederkehrende Arbeiten
Der Benutzer oder die Benutzerin kann das Intervall der wiederkehrenden Aufgaben bearbeiten. Beim Bearbeiten wird definiert, ob eine Aufgabe einmalig ist oder mehrfach ausgeführt werden soll. Dabei gibt der Benutzer oder die Benutzerin an, in welchen Intervallen diese auftreten wird. Die Intervalle werden in Tagen angegeben.

Bei Änderung von bestehenden Aufgaben, wird die Aufgabenliste angepasst.

UC 3.3: Verwalte den Status der Aufgaben und sende Benachrichtigungen für fällige Aufgaben

Der Benutzer oder die Benutzerin gibt in der Aufgabenübersicht eine Rückmeldung, wenn die Aufgaben erledigt sind. Diese werden dann in der Planung nicht mehr angezeigt. Täglich werden Benachrichtigungen für die Aufgaben versendet, die an diesem Tag fällig sind.

2.1.4 Use-Case-Gruppe 4

UC 4.0 Wetterprognose für den eingestellten Standort abfragen
Der Benutzer oder die Benutzerin kann einstellen, wo sich ein Garten befindet (Adresse des Gartens). Der Standort bleibt so lange gespeichert, bis die Einstellung wieder vom Benutzer oder Benutzerin geändert wird oder der Garten gelöscht wird.
Der Benutzer oder die Benutzerin kann das prognostizierte Wetter der nächsten Tage für den Standort des Gartens ansehen.

UC 4.1 Wetterbedingte wiederkehrende Aufgaben überspringen
Wiederkehrende Aufgaben, die vom Wetter abhängig sind, können automatisch von der Applikation übersprungen werden, wenn diese Aufgaben aufgrund der Wetterlage hinfällig werden. Es kann die Bewässerung bei prognostiziertem Regen übersprungen werden.

UC 4.2 Wetterbedingte Aufgaben generieren
Einmalige Aufgaben, die vom Wetter abhängig sind, können automatisch von der Applikation generiert werden, wenn diese speziell aufgrund der Wetterlage benötigt werden. Es können spezielle Massnahmen zum Schutz der Pflanze empfohlen werden, wenn Frost prognostiziert wird.

2.2 Zusätzliche Anforderungen

2.2.1 Einfache Bedienung

Die allgemeine Bedienung der Applikation soll so einfach wie möglich und selbsterklärend sein. Nutzer:innen sollen alle Funktionalitäten nutzen und verstehen können, ohne eine Anleitung herbeizuziehen. Alle Menüs und Ansichten in der Applikation sollen einheitlich aussehen und gleich funktionieren. Wenn die Nutzer:innen ein bestimmtes Verhalten erwarten, soll dies auch entsprechend funktionieren. Für unerfahrene Nutzer:innen soll es möglich sein, innerhalb von maximal zehn Minuten einen Garten zu erstellen, dessen Standort festlegen, erste Pflanzen einplanen und auch individuelle Aufgaben für die geplanten Pflanzen zu erstellen.

2.2.2 Individualisierbarkeit

Unerfahrene Nutzer:innen benötigen andere Funktionalitäten als erfahrene. Die Applikation ist übersichtlicher, wenn nicht benötigte Funktionen ausgeblendet werden können. Aus diesem Grund sollen mindestens das Tutorial, die Standorte der Gärten, sowie die Wetteranbindung deaktiviert und ausgeblendet werden können.

2.2.3 Datenspeicherung

Alle von den Nutzer:innen gespeicherten Daten sollen direkt abgespeichert werden. Bei unerwartetem Beenden der Anwendung gehen keine Daten verloren und stehen nach einem Neustart der Applikation wieder zur Verfügung.

Alle lokal gespeicherten Daten der Applikation sollen in einem Verzeichnis gespeichert werden. Die nutzenden Personen können so einfach eine persönliche Datensicherung machen, indem sie eine Kopie dieses Verzeichnisses erstellen.

2.2.4 Reaktionsgeschwindigkeit

Wenn ein Nutzer oder eine Nutzerin in der Applikation etwas anklickt, sollen keine Wartezeiten entstehen die länger als 500 Millisekunden andauern. Aufwändigere Operationen wie zum Beispiel das Senden von E-Mails sollen im Hintergrund geschehen und für die Nutzer:innen nicht spürbar sein.

2.2.5 Fehlerbehandlung

Sollten aus beliebigen Gründen Fehler auftreten, wie zum Beispiel fehlende Internetverbindung oder fehlende Datenbankdatei, darf die Applikation nicht unerwartet stoppen. Stattdessen sollen für die Nutzer:innen verständliche Fehlermeldungen angezeigt werden, sodass diese das Problem selbst beheben können oder wissen, wem der Fehler zu melden ist.

2.3 Domänenmodell

Im folgenden Modell wird die Problemdomäne des Projekts beschrieben (siehe Abbildung 5).

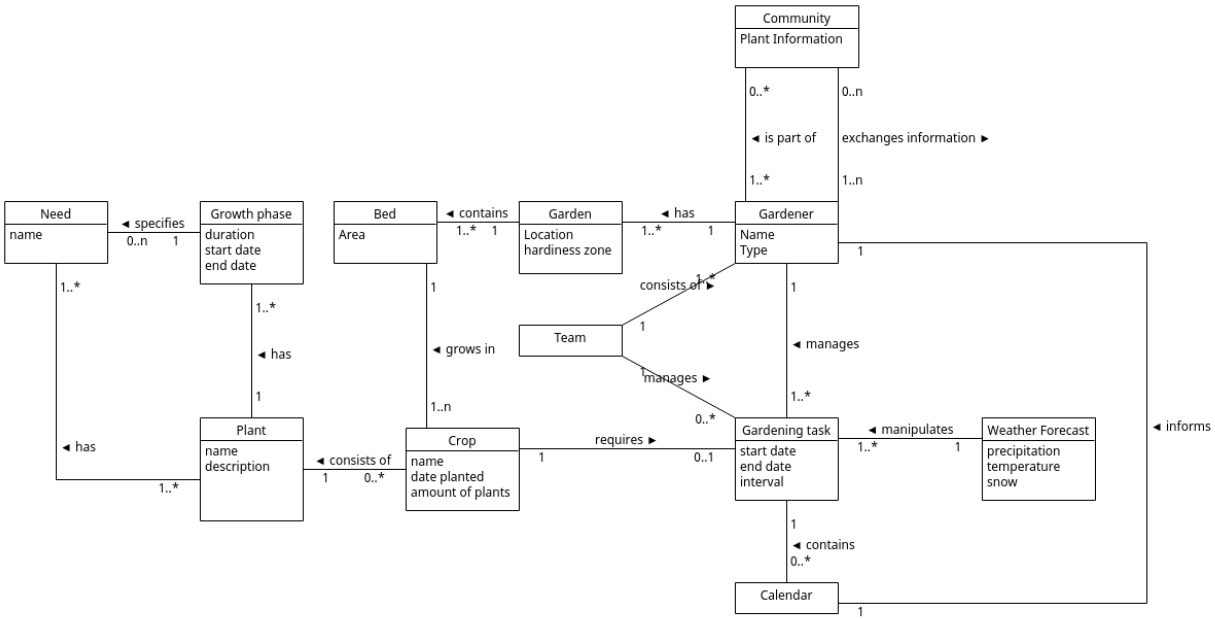

Abbildung 5: Domänenmodell Diagramm

Die Problemstellung befasst sich mit der Planung von Gärten. Hauptakteure sind somit die Gärtner:innen, welche diese Planung durchführen. Diese können Teil eines Teams oder allein sein. Sie haben mindestens einen Garten, welcher aus einem oder mehreren Beeten besteht. Pflanzen haben verschiedene Bedürfnisse und mehrere Wachstumsphasen. Anpflanzungen (im Diagramm “Crop”) wachsen in einem Beet und umfassen ein oder mehrere Exemplare einer Pflanzenart. Mit Anpflanzungen sind zeitlich fixierte Aufgaben verbunden, welche zur erfolgreichen Ernte erforderlich sind. Wetterbedingungen, wie Niederschlag oder Temperatur können einen Einfluss auf die Aufgaben haben. Zum Beispiel muss bei viel Niederschlag weniger getränkt werden und bei Frost müssen zusätzliche Massnahmen ergriffen werden. Weitere Aufgaben, die im Unterhalt eines Gartens anfallen, werden von den Gärtner:innen oder den Teams eingeplant.
Alle, die in einem Garten Gemüse anbauen und ihr Wissen mit anderen teilen, bilden eine Community. Sie können auch mehrere Communities bilden, zum Beispiel über verschiedene Kommunikationsmittel oder in unterschiedlichen Regionen. Innerhalb dieser Communities werden Informationen ausgetauscht, die schwierig in Büchern zu finden wären.

3 Design

Im Folgenden wird das technische Design der Software beschrieben.

3.1 Softwarearchitektur

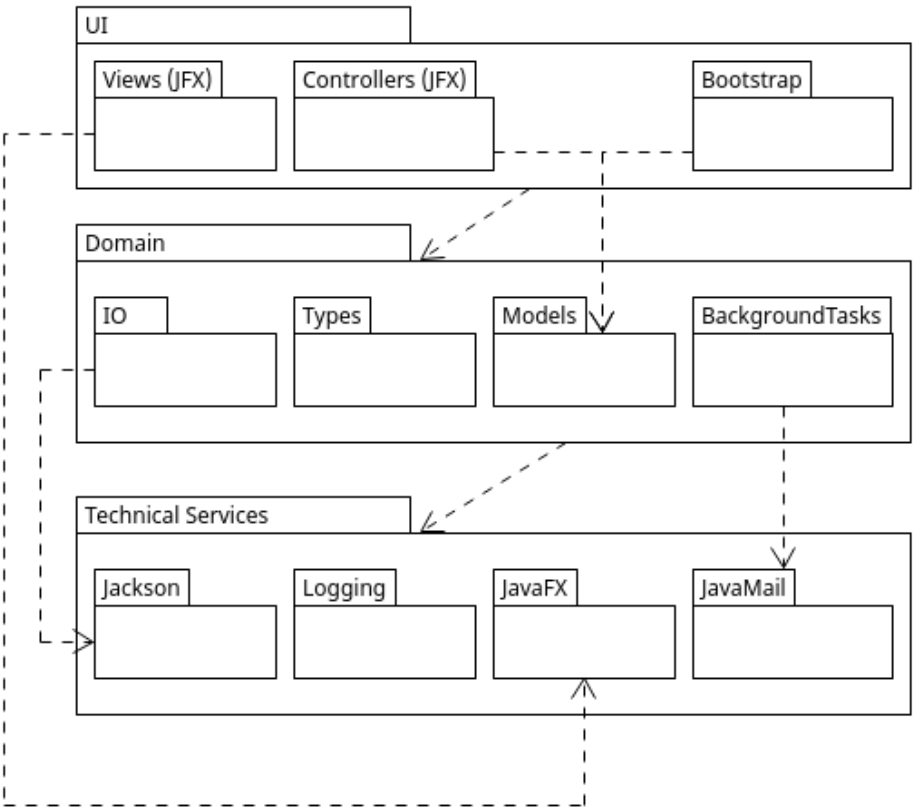

Abbildung 6: Schichten-Diagramm

Die Architektur der Software ist nach dem Model-View-Controller-Pattern aufgebaut. Die Aufteilung der Pakete ist im Schichten-Diagramm dargestellt (siehe Abbildung 6).

3.1.1 UI-Schicht

Für die Umsetzung der grafischen Benutzeroberfläche wird das JavaFX-Framework verwendet. In diesem Kontext sind die Views als einfache Beschreibung der grafischen Elemente in der Form von XML-Dateien zu verstehen. Controller sind dafür zuständig Eingaben zu verarbeiten und die Views entsprechend zu aktualisieren.

Das Bootstrap-Paket dient dazu, Views korrekt zu laden und, wo möglich, zu cachen. In diesem Schritt ist es auch nötig den zugehörigen Controllern die Models einzuspeisen, von denen sie abhängen.

3.1.2 Domänen-Schicht

“Types” repräsentieren die Eigenschaften der grundlegenden Datentypen der Software und bieten Funktionalität zur Ermittlung von mit ihnen verbundenen Daten. Dies wurde so gewählt, damit die anderen Teile der Domänenlogik mit einheitlich strukturierten Daten arbeiten können.

Klassen des IO-Pakets sind Schnittstellen zwischen externen Datenquellen, wie Dateien oder Datenbanken, und den Datentypen des “Types”-Pakets. Das Herzstück bilden die Schnittstellenbeschreibungen der drei Datenquellen. Die PlantList liefert statische Daten zu Pflanzen, die CropList liest und

speichert die von den Nutzer:innen gewählten Anpflanzungen und die TaskList liest und speichert die Aufgabenliste.

Das Paket enthält ausserdem je eine Implementation der Schnittstellen, welche JSON-Dateien als externe Datenquellen verwenden. Diese verwenden die Bibliothek “Jackson”. Sie wurde gewählt, da sie aus JSON-Konstrukten direkt die Klassen des “Types”-Pakets generieren kann, was bei allfälligen Änderungen der Datenstruktur viel Entwicklungsaufwand spart.

Alternative externe Datenquellen können zukünftig als Implementation der Schnittstellen hinzugefügt werden, ohne Änderungen an anderen Teilen der Software zu erfordern.

Datenquellenunabhängige Funktionalität, wie zum Beispiel das Filtern der Daten, ist der Aufgabenbereich des “Models”-Pakets. Dadurch wird Code-Duplizierung vermieden. Models enthalten die Daten, die in einer View dargestellt werden sollen und stellen die darauf anwendbaren Operationen zur Verfügung.

BackgroundTasks sind wiederkehrende Hintergrundaufgaben. Dazu gehören das Überprüfen der anstehenden Aufgaben in der Aufgabenliste und das allfällige Auslösen entsprechender Benachrichtigungen, sowie die Aktualisierung der Wetterdaten aus einer API.

3.1.3 Technical Services-Schicht

Wie bereits erwähnt, wird die Jackson-Bibliothek zur Deserialisierung von JSON-Daten eingesetzt und JavaFX zum Aufbau der Grafischen Benutzeroberfläche.

Für das Schreiben von Applikations- und Fehlerprotokollen wird der Logging-Dienst der Java Standardbibliothek eingesetzt. Für den aktuell geplanten Applikationsumfang ist dieser ausreichend. Dieses Paket wird von Paketen der Domänen-Schicht, sowie von den Controllern verwendet, weshalb die Pfeile im Diagramm weggelassen wurden.

Zum Senden von E-Mail-Benachrichtigungen wird die Bibliothek JavaMail eingesetzt. Diese erfüllt die Anforderungen der Software und erfordert wenig zusätzlichen Entwicklungsaufwand.

3.2 Design-Klassendiagramm

Die Abbildung 7 zeigt das Design-Klassendiagramm der Software. Dieses enthält alle Klassen, die zur Domänenlogik gehören.

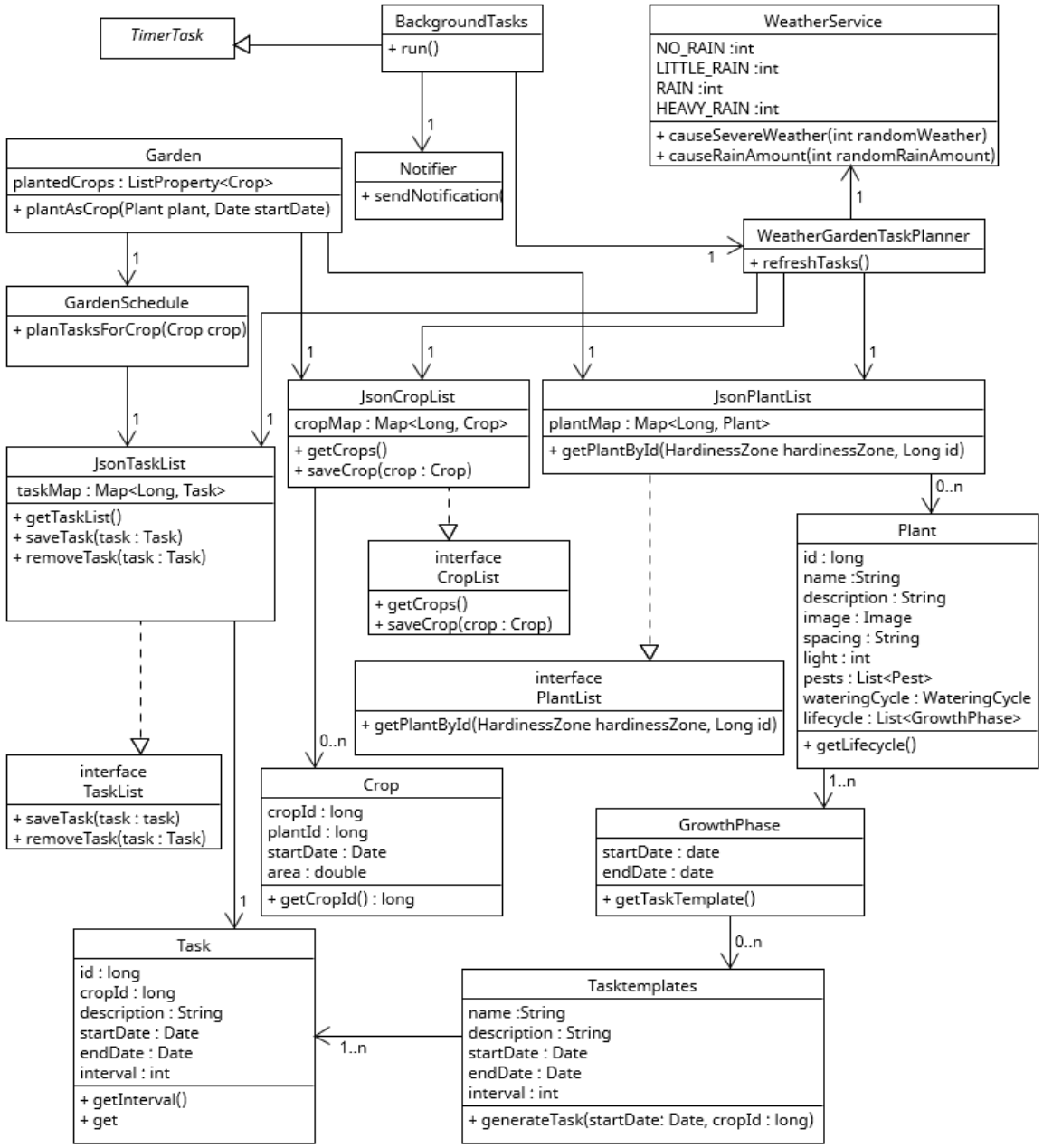

Abbildung 7: Design-Klassendiagramm Entwurf

3.3 Angewandte Design-Patterns

Im Folgenden ist beschrieben, wie gängige Design-Patterns im Design der Software angewandt wurden.

3.3.1 Dependency-Injection

Zwei Arten von Dependency-Injection werden in der Applikation verwendet. Die Klassen des “Models”-Pakets benötigen Zugriff auf die geladenen Daten, welche durch Klassen des IO-Pakets zur Verfügung gestellt werden. Um Konflikte durch mehrfaches Laden von Daten zu vermeiden, werden PlantList, TaskList und CropList-Objekte durch Konstruktor-Argumente übergeben.

Für die Klassen des “Controllers”-Pakets ist dieser Ansatz nicht geeignet, da diese vom JavaFX-Framework instanziiert werden. Hier werden daher die in Java verfügbaren Reflection-Techniken eingesetzt, um die Abhängigkeiten nach der Erzeugung des Objekts dynamisch einzufügen.

3.3.2 Strategy-Pattern

Für die Klassen des IO-Pakets wurden Schnittstellen definiert, die es erlauben, unterschiedliche Persistenz-Methoden anzubinden.
Am Beispiel der CropList-Schnittstelle (Abbildung 8) ist zu erkennen, dass die aktuell verwendete Implementation “JsonCropList”, welche die Daten in eine JSON-Datei auf dem ausführenden System speichert, ersetzt werden kann, ohne dabei Änderungen an den verwendenden Klassen vornehmen zu müssen. Die tatsächlich verwendete Strategie wird dann mittels der oben beschriebenen Dependency-Injection an die gesamte Applikation verteilt.

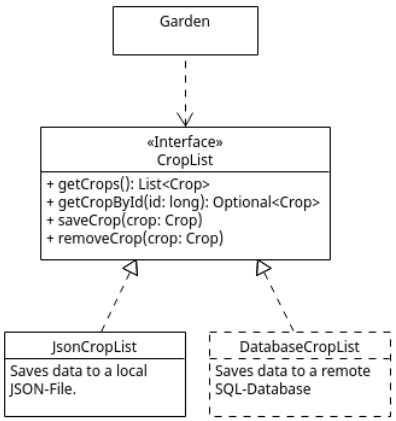

Abbildung 8: Strategy Pattern

3.4 Kommunikationsdiagramm plantAsCrop

Im folgenden Kommunikationsdiagramm wird die genaue Abfolge der Aufrufe, die bei Ausführung der Systemoperation plantAsCrop() erfolgen, dokumentiert.

Als erstes wird die Pflanze, die angepflanzt wird, als Crop in den GradenSchedule gespeichert. Danach wird aus der Plant, welche die statischen Eigenschaften einer Pflanze beschreibt, anhand eines Anpflanzdatums (startDate) eine Nutzpflanze (Crop) erstellt. Diese wird in der JsonCropList gespeichert. Danach werden alle Tasks geplant, die an dieser Pflanze ausgeführt werden müssen, bis sie geerntet werden kann. Dazu holt die GardenSchedule-Klasse die Templates für die Tasks aus dem Pflanzenobjekt und erstellt die Tasks relativ zum gegebenen Anpflanzdatum. Diese Tasks speichert sie in die JsonTaskList (siehe Abbildung 9).

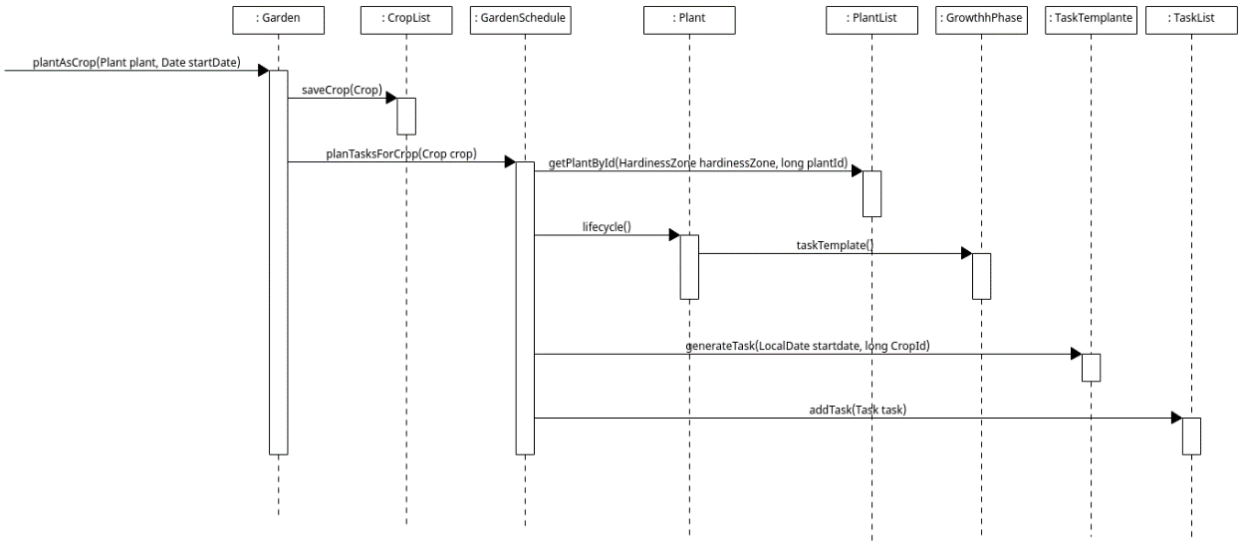

Abbildung 9: Kommunikationsdiagramm der Systemoperation plantAsCrop

3.5 Kommunikationsdiagramm run() BackgroundTask

Im folgenden Diagramm (siehe Abbildung 10) ist der Ablauf der Funktion im Hintergrund der Applikation dargestellt. Die run()-Methode wird täglich von einer “Timer”-Klasse aufgerufen. Dabei werden laufend die Wetterdaten erneuert. Sie werden auf Anzeichen von Hagel, Frost und Schnee überprüft und es werden dafür nötige Tasks erstellt. Dabei wird überprüft, ob es für das Datum, an dem der nötige Task eingefügt werden soll, schon ein solcher Task existiert. Falls ein schon einer existiert, wird der neue Task nicht hinzugefügt.
Danach wird die Regenmenge abgefragt, die in den letzten sieben Tagen gefallen ist. Es wird geprüft, ob die Regenmenge die erforderliche Regenmenge der Pflanze überschritten wird. Wenn das der Fall ist, wird die Bewässerungsaufgabe («water plant») um das vordefinierte Intervall der Bewässerung aufgeschoben.

Die “Notifier” Klasse haben wir nicht in das Diagramm eingefügt. Die Applikation generiert pro Task eine E-Mail mit den in der Config definierten Nutzerdaten.

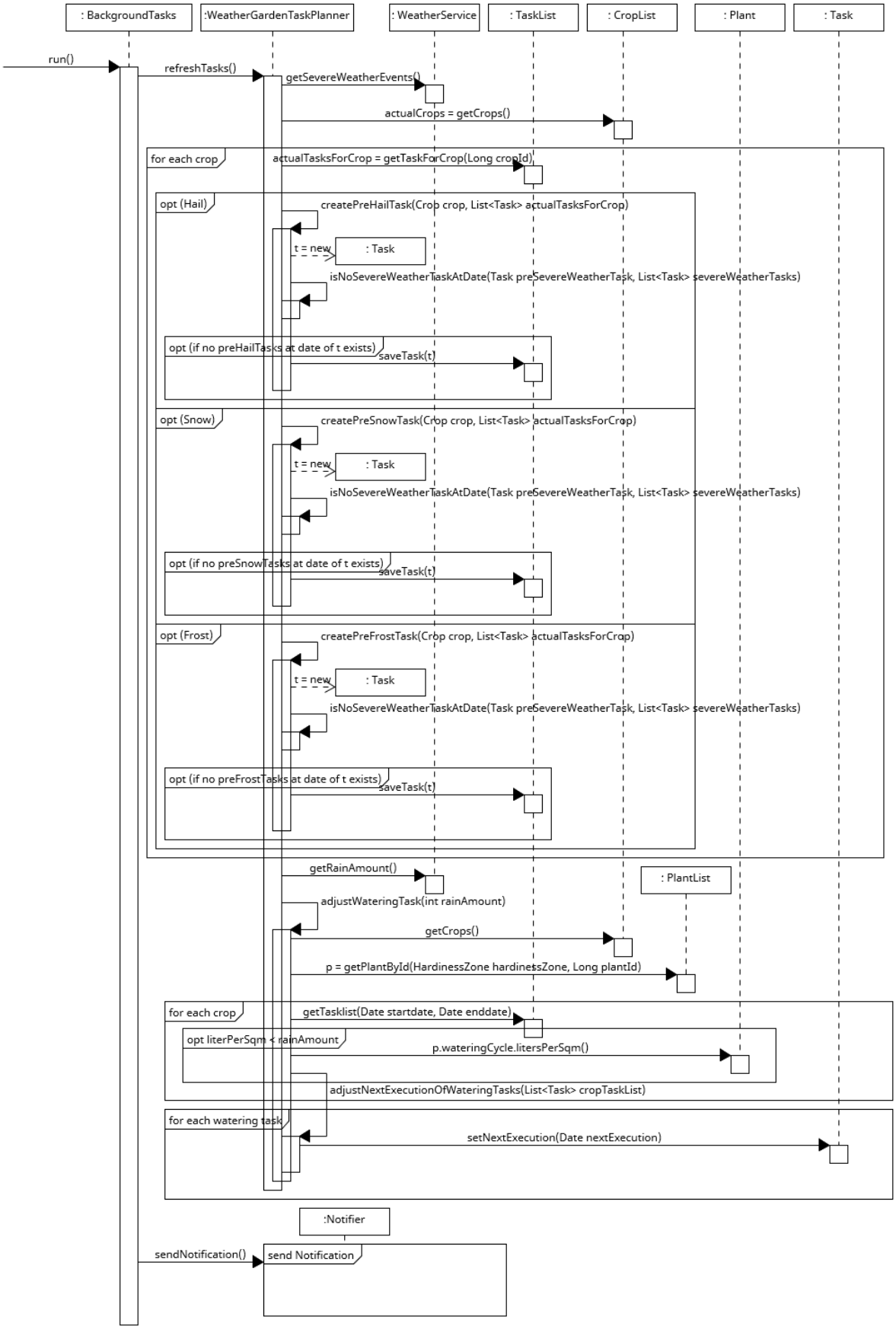

Abbildung 10: Kommunikationsdiagramm der Systemoperation run Backgroundtask

4 Implementation

In diesem Kapitel werden Details der Implementation beschrieben. Der Fokus liegt auf der Verifizierung des Codes und der Architektur.

4.1 Lieferergebnisse

Für die Betaphase wird ein Zip-Archiv zur Verfügung gestellt.
Dieses beinhaltet neben den Java-Bibliotheken auch Start-Scripts für die jeweiligen Plattformen.
Hinzu kommt eine einfache Installationsanleitung und ein “getting started”.
Der Code inklusive Gradle-Scripts werden auf Github zur Verfügung gestellt.
Die Testergebnisse befinden sich im Anhang dieses Dokuments.

4.2 Teststrategie

Zur Entwicklung wird die “Code-Driven Development”-Strategie verfolgt, wobei zur Parallelisierung des Arbeitsprozesses gleichzeitig Tests und Implementation umgesetzt wurden. Innerhalb eines Sprints wird für jede Anforderung mindestens ein automatisierter Test mit dem “JUnit 5” Framework erstellt, so dass die “Definition of Done” nachgewiesen werden kann. Im Projektverlauf wurde im Team entschieden, auf welche Komponenten zusätzlicher Testfokus gelegt werden soll.

4.2.1 Testumgebung

Die automatisierten Tests werden mittels des Gradle-Build-Systems ausgeführt. Die dazu benötigten Testdateien sind im Repository integriert. Die Abdeckung der Tests wird durch das “Coverage”-Plugin der IntelliJ-Entwicklungsumgebung festgestellt. Eine detailliertere Auswertung liefert das Plugin Jacoco.

Die Grafische Oberfläche wird getestet, indem von Hand Aktionen im UI ausgeführt werden. Dazu gibt es anhand der Use-Cases definierte Abläufe, die durchlaufen werden müssen.

4.2.2 Modulebene

Auf Modulebene wurden Tests zum einen als Black-Box-Tests aufgrund der Anforderungen geschrieben und nach der Implementation der Klassen, und, wo notwendig, mit White-Box-Tests ergänzt oder zum anderen gleichzeitig zur Kodierung von den Entwicklern selbst geschrieben. Sie standen damit früh für jedes tägliche Inkrement und auch als Regressionstests bei Umbauten zu Verfügung. Durch unterschiedliche Urheber von Implementation und Tests konnten an einer Stelle auch Unklarheiten bei der Formulierung der Funktionen und ihrer Parameter entdeckt werden.

Durch Modultests werden 100% der öffentlichen Schnittstellen abgedeckt.

4.2.3 Integrationstests

Derzeit gibt es nur Integrationstests für die Klassen des IO-Pakets, wo die richtigen Klassen der Jackson-Bibliothek zum Lesen richtiger Dateien eingesetzt werden.

Für andere Teile der Applikation konnten bisher noch keine Integrationstest definiert werden, da die Schnittstellen und das Zusammenspiel der Module noch nicht final definiert waren. Im nächsten Schritt werden Integrationstest erstellt für jene Module, deren Zusammenspiel geklärt ist. Die Tests werden stufenweise implementiert, wobei auf jeder Stufe ein zusätzliches Modul mitgetestet wird. Nur die auf einer Stufe neu hinzugekommenen Funktionen sollen jeweils geprüft werden.
Mit automatisierten Tests werden 100% der für die UC notwendigen, Inter-Modularen Szenarien abgedeckt.

4.2.4 Systemtests

Die Systemtests wurden von zwei Entwicklern unabhängig voneinander anhand der UC 1 und 2, welche bereits teilweise implementiert wurden, im UI als Black-Box-Test durchgeführt und verifizieren so die Systemarchitektur.

100% der Use-Cases und so viele der nichtfunktionalen Anforderungen wie möglich werden dabei abgedeckt.

4.2.5 Fehlermanagement

Fehler werden vom jeweils Feststellenden in den GitHub-Issues festgehalten. Die Vergabe der Zuständigkeit für die Behebung erfolgt in den regelmässigen Team-Sitzungen.

4.3 Verifikation Systemarchitektur

In diesem Unterkapitel wird beschrieben, wie die Systemarchitektur der Software verifiziert wurde.

4.3.1 UC 1 Anpflanzungsauswahl (abgedeckt & implementiert)

UC 1.0 und 1.1 können direkt nach dem Start des Programmes mit einem Klick auf den Button “Add new Plant” verifiziert werden: Die Optionsfelder in der rechten Seitenleiste ermöglichen die Einschränkung der angezeigten Pflanzen nach Jahreszeit oder Klimazone.

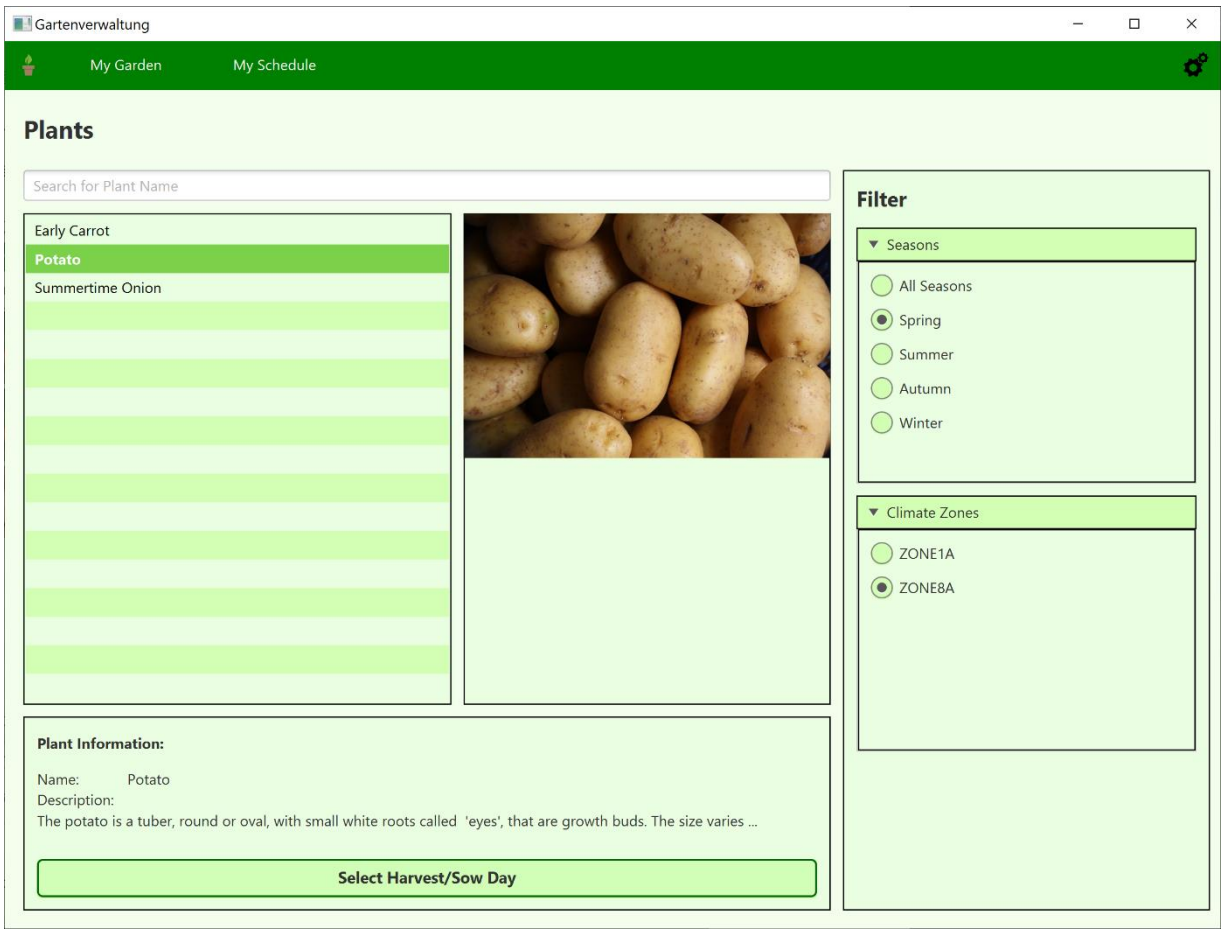

Abbildung 11: Anpflanzungsauswahl

4.3.2 UC 2 Pflanzeninformation (abgedeckt)

Beim Klick auf eine Pflanze kann UC 2.0 verifiziert werden, wobei die Detailinformationen zu Wasserbedürfnissen derzeit noch nicht dargestellt werden. (Aktueller Stand: Abbildung 11)

Nach Klick auf eine Pflanze kann UC 2.1 verifiziert werden, indem der “Select Harvest/Sow Day”-Knopf gedrückt wird. Dabei erscheint eine Datumsauswahl, in welcher nur mögliche Pflanz- oder Erntedaten wählbar sind (Abbildung 12). Nach Bestätigung des Datums erscheint ein entsprechender Eintrag im Reiter “My Garden”.

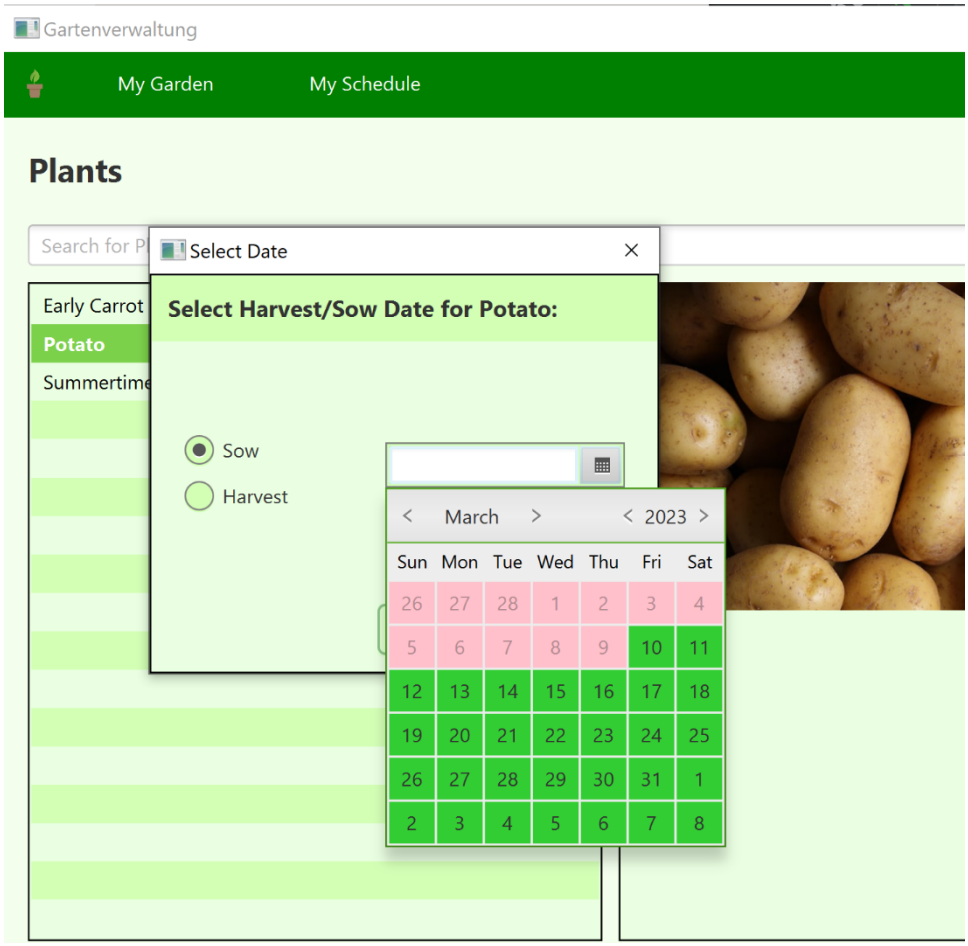

Abbildung 12: Datenauswahl

4.3.3 UC 3 Aufgabenliste

UC 3.0: Nach Auswahl der Pflanze (UC 2.1) wird eine Reihe von Aufgaben zu deren Anbau erstellt. Die Aufgaben werden unter dem Reiter “My Schedule” in einer kalenderartigen Liste angezeigt.
UC 3.1: Die Aufgabenliste aus 3.0 kann über die grafische Oberfläche bearbeitet werden. Neue Aufgaben können hinzugefügt werden und bestehende können gelöscht werden.
UC 3.2: Eine Aufgabe kann als wiederholend eingerichtet werden. Das Wiederholungsintervall wird dazu in Tagen angegeben.

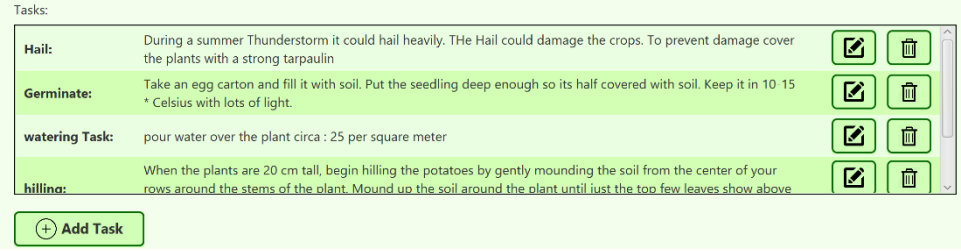

Abbildung 13: Aufgabenliste

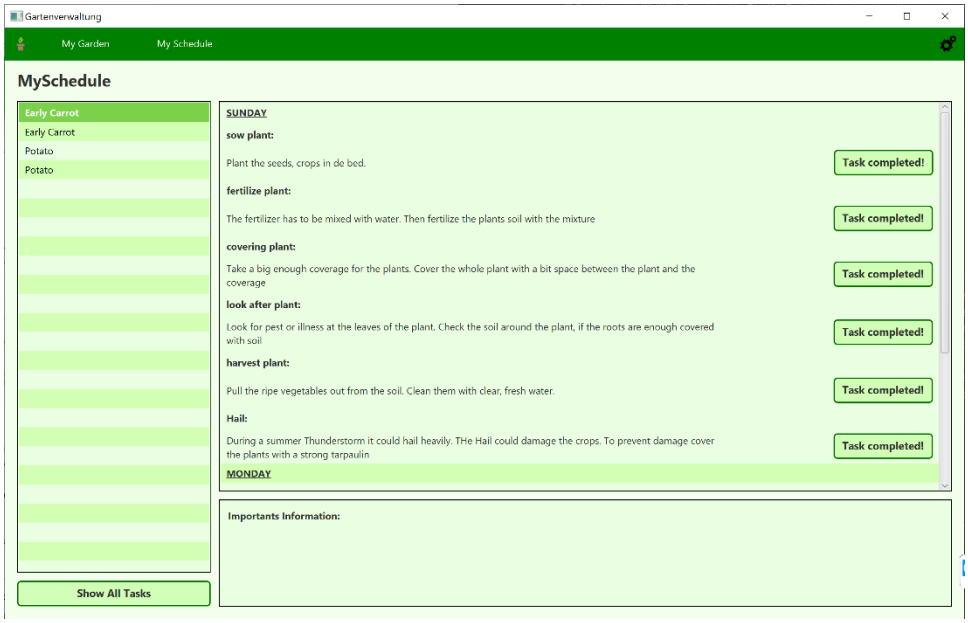

Abbildung 14: My Schedule

4.3.4 UC 4 Gartenstandort und Wetterprognose (teilweise implementiert)

UC 4.0: Der Gartenstandort und die Klimazone können in den Einstellungen mit dem Button “Set Location” bzw. Popup Hardiness Zone und seine Grösse mit “Set Area” bei der jeweiligen Bepflanzung erfasst und geändert werden.

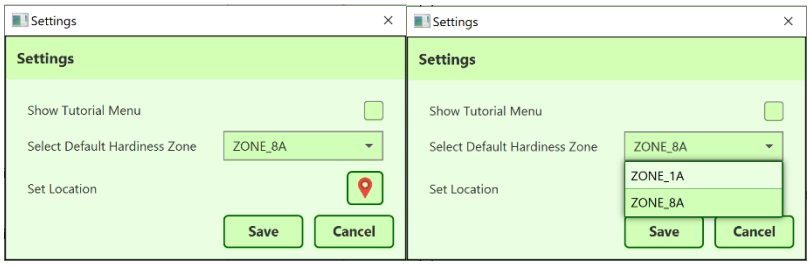

Abbildung 15: Set Location & Hardiness Zone

UC 4.1: Für den Standort kann die Wetterprognose abgefragt werden (heute und nächste Tage). In der Beta-Version wird eine fixe Prognose implementiert, um die Funktionalität wiederholbar testen zu können.

UC 4.2: Die Wetterprognosen können auf die Taskliste angewandt werden und entfernen überflüssige Aufgaben.

4.3.5 UC 5 Schädlinge

UC 5.0: Schädlinge zu einer Pflanze werden direkt unten auf der Detailseite der Pflanzeninformationen dargestellt. Dazu werden direkt auch mögliche Massnahmen angezeigt.

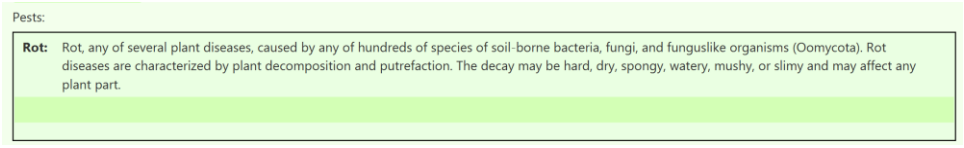

Abbildung 16: Schädlingsinformation

UC 5.1: Ein tatsächlich aufgetretener Schädlingsfall kann vermerkt und die Behandlung in die Taskliste übernommen werden.

4.3.6 Weitere Anforderungen

Mehrfachfilterung: In der Pflanzenliste sollen die angezeigten Einträge nach mehreren Kriterien gleichzeitig eingeschränkt werden können (noch nicht implementiert).
Variables Layout: Einzelne Bereiche der Grafischen Oberfläche sollen auf Verlangen ausgeblendet werden können (noch nicht implementiert).
Antwortzeiten: Durch die grafische Oberfläche ausgelöste Aktionen sollen möglichst schnell abgeschlossen sein. Um dies zu verifizieren, sollen die Ausführungszeiten der entsprechenden Integrationstests gemessen werden mit dem “Stopwatch”-Modul von JUnit.

5 Resultate

Hier wird beschrieben, welche Ziele erreicht worden sind, welche noch offenstehen und was für Weiterentwicklungsmöglichkeiten es gibt

5.1 Erreichte Ziele

Die Ziele, welche wir bisher erreicht haben, sind:

- Die Grundfunktionalitäten des Produktes sind vorhanden. Dies beinhaltet, dass ein Datensatz von Pflanzen nach Jahreszeiten und Klimazone gefiltert werden kann und anhand von Anpflanz- oder Erntezeit geplant werden kann, wann diese gepflanzt werden sollen. Dabei werden die nötigsten Informationen dargestellt und die fälligen Aufgaben für die Pflege werden dem Benutzer oder Benutzerin angezeigt.
- Der Benutzer oder die Benutzerin kann die Liste der vorhandenen Aufgaben bearbeiten oder mit eigenen Aufgaben erweitern.
- Der Benutzer wird über Änderungen von neu anfallenden oder abgeänderten Aufgaben informiert oder warnt von noch nicht erledigten Aufgaben.
- Die Aufgabenliste wird anhand der Wetterprognosen abgeändert und informiert den Benutzer oder die Benutzerin.
- Die Anwendung kann einfach navigiert werden und dessen Funktionalitäten sind für die Benutzer und Benutzerinnen selbsterklärend und werden auch beschrieben in der Applikation gezeigt, wie diese funktionieren.

5.2 Offene Punkte

Die Ziele, welche wir bisher noch nicht erreicht haben, sind:

- Im Moment werden die grössten Schädlinge angegeben, jedoch können die Pestizide noch nicht abgerufen werden, damit Aufgaben erstellt werden. Diese werden im nächsten Release hinzugefügt.
- Wegen Lizenzproblemen von den Wetterdienst APIs kann das Wetter im Moment nur limitiert abgerufen werden. Anfragen an verschiedene Anbieter wurden gestellt, jedoch werden diese sich nicht vor dem Beta-Release zurückmelden. Es wird in den folgenden Wochen eingebunden.
- Die Applikation kann zu dieser Zeit nicht definieren, wie gross ein Garten ist und wie viel Abstand zwischen den verschiedenen Pflanzen nötig ist. Dies wird ebenfalls im nächsten Release mit einer interaktiven Benutzeroberfläche hinzugefügt.

5.3 Ausblick auf Weiterentwicklung

Aspekte, welche weiterentwickelt werden können, sind:

- Die Benutzer und Benutzerinnen können mehrere Anbauflächen/Gärten definieren, wo mehr für Grossbauern/-innen ausgerichtet sind, welche mehr Flächen als nur ein Garten abdecken müssen.
- Die Möglichkeit, dass mehrere Personen im gleichen Garten gemeinsam arbeiten können. Dabei wird für beide Benutzer/-innen die Pflanzen und deren Aufgaben angegeben und kann von jedem erledigt werden. Als Beispiel ein gemeinsamer Garten von zwei Nachbarn.
- Eine Online-Community-Einbindung, dass ein Forum erstellt wird, wo alle Benutzer und Benutzerinnen sich gegenseitig austauschen können. Auch sie dort ihre eigens definierten Pflanzen und Aufgaben teilen, dass andere diese abrufen und in deren Anwendung verwenden können.

6 Anhang

Im Anhang sind zusätzliche Informationen über das Projekt beigelegt. Dies beinhaltet das Projektmanagement, die Testresultate und eine Installations- und Gebrauchsanweisung für die Software.

6.1 Projektmanagement

Der Aufwand der Aufgaben der Iteration (siehe Tabellen 4, 7, 10, 13, 16 und 19) werden nach geplanten und aktuellen Stunden dargestellt. Dazu wird noch angegeben, welches Mitglied für diese Aufgabe verantwortlich war und ob diese erfüllt worden sind. Dabei kann es auch auftreten, dass manche Aufgaben nur teilweise (TW) erfüllt worden sind. Für die Verantwortlichen wurde das Kürzel des Namens der ZHAW verwendet (siehe E-Mail-Adressen auf dem Deckblatt).

Wegen fehlenden Zeiteinträgen für die Aufgaben der ersten vier Iterationen können diese leicht von den eigentlichen Ergebnissen abweichen. Der totale Zeitaufwand pro Iteration, wurde jedoch korrekt übertragen.

Die Massnahmen (siehe Tabellen 5, 8, 11, 14, 17 und 20) der Iteration beziehen sich auf die nicht erfüllten oder nur teilweise erfüllten Ziele und weitere Entscheidungen, welche während der Iteration vorgenommen worden sind.

Die Planung der nächsten Iteration (siehe Tabellen 6, 9, 12, 15 und 18) wird in verschiedene Kategorien aufgeteilt.

6.1.1 Iteration 1

Aufwand:

Tabelle 4: Aufwand Iteration 1

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Erstellung der Projektskizze	20	18	gulerdav, huttegia	Ja
2	Anforderungen für das Projekt wurde erstellt	6	8	schrom01	Ja
3	Risiken wurden für das Projekt analysiert	4	6	csomoeli	TW
4	Grober Grundsatz der Use-Cases wurden erstellt	8	8	giavaphi	Ja
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	20	20	Alle Mitglieder	-
	Total	**60**	**62**		

Massnahmen:

Tabelle 5: Massnahmen Iteration 1

Massnahmen nicht/teilweise erfüllte Ziele:
Risiken wurden für das Projekt analysiert: Bei der Risikoanalyse haben wir zu wenig Zeit investiert und nicht vollständig ausgefüllt. Im ersten Meeting nach der Iteration sind die fehlenden Risiken gefunden und analysiert worden. Aus allen analysierten Risiken ist eine Risikoliste erstellt worden.
Weitere Massnahmen (erfüllte Ziele):
Grober Grundsatz der Use-Cases wurden erstellt: Die Use-Cases sind nicht stark genug aufgeteilt worden. In einem Meeting nach der ersten Iteration haben wir entschieden die momentanen Use-Cases weiter aufzuteilen und anhand der alten Use-Cases zu gruppieren. (Beispiel: 1.0, 1.1, 2.0, 2.1, …)

Weitere Massnahmen:
-

Planung nächste Iteration:

Tabelle 6: Planung für Iteration 2

Design:
Erster Entwurf des Domänenmodelles.
Erster Entwurf des Klassendiagramms.
Use-Case von 1.0 bis 2.1 werden ausformuliert. (brief/casual)
GUI:
User Interface des Produktes wird skizziert.
Navigation zwischen den Menüeinträgen wird mithilfe von FXML-Dateien erstellt.
Implementation:
Record Klasse für die Pflanzen wird erstellt.
IO:
Die JSON-Datenbankstruktur wird erstellt.
Adapter Klasse für Pflanzenliste wird erstellt (kann die Daten der JSON-Datei in eine Liste von Pflanzen-Objekte umwandeln).
Weitere Ziele:
Anforderungen werden abgegrenzt, welche für den Prototypen und für den Beta-Release realisiert werden.

6.1.2 Iteration 2

Aufwand:

Tabelle 7: Aufwand Iteration 2

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Erster Entwurf des Domänenmodelles	5	5	huttegia	Ja
2	Erster Entwurf des Klassendiagramms	8	8	huttegia	Ja
3	Use-Case von 1.0 bis 2.1 werden ausformuliert (brief/casual)	4	5	schrom01	Ja
4	User Interface des Produktes wird skizziert	4	4	giavaphi	Ja
5	Navigation zwischen den Menüeinträgen wird mithilfe von FXML-Dateien erstellt	6	6	giavaphi	TW
6	Record Klasse für die Pflanzen wird erstellt	8	7	schrom01	Ja
7	Die JSON-Datenbankstruktur wird erstellt	5	5	gulerdav	Ja
8	Adapter Klasse für Pflanzenliste wird erstellt	8	8	gulerdav	TW
9	Anforderungen werden abgegrenzt, welche für den Prototypen und für den Beta-Release realisiert werden	4	4	csomoeli	Ja
10	Testfälle für die Adapter Klasse der Pflanzenliste werden erstellt	6	6	csomoeli	Ja
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	40	40	Alle Mitglieder	-
	Total	**100**	**100**		

Massnahmen:

Tabelle 8: Massnahmen Iteration 2

Massnahmen nicht/teilweise erfüllte Ziele:
Navigation zwischen den Menüeinträgen wird mithilfe von FXML-Dateien erstellt: Anhand von falschen Konfigurationen der IDE konnten die FXML-Dateien nicht im Projekt ausgeführt werden und mussten in einem separaten Projekt erstellt werden, um implementiert zu werden. Nach der Iteration wurde dies behoben und dem Projekt hinzugefügt.
Adapter Klasse für Pflanzenliste wird erstellt: Die Adapter-Klasse erstellt eine Liste von Pflanzen Objekten, welche jedoch noch keine Bilder beinhalten. Dies wurde in der nächsten Iteration behoben.
Weitere Massnahmen (erfüllte Ziele):
Testfälle für die Adapter Klasse der Pflanzenliste werden erstellt: Die Testfälle für die Adapter Klasse wurde für die nächste Iteration geplant, jedoch wurden diese schon in dieser Iteration erstellt.
Weitere Massnahmen:
-

Planung nächste Iteration:

Tabelle 9: Planung für Iteration 3

Design:
Use-Cases von 3.0 bis 3.2 werden ausformuliert. (fully-dressed/casual)
GUI:
Eine Liste von Pflanzen können im GUI abgebildet werden.
Einzelne Pflanzen können ausgewählt werden und genauere Information können abgebildet werden.
Bilder der Pflanzen werden abgebildet, wenn diese ausgewählt worden sind.
Die Pflanzenliste kann nach Saison, Klimazone und Suchanfrage gefiltert werden.
Implementation:
Model für die Pflanzenliste wird erstellt und mit dem Controller verbunden.
Testfälle für das Model der Pflanzenliste werden erstellt.
IO:
Adapter-Klasse für die Liste der geplanten Pflanzen (Crop) wird erstellt. (von JSON-Datei zu Liste von Crops, anhand der Crop-Klasse)
Adapter-Klasse für die Liste der Aufgaben wird erstellt. (von JSON-Datei zu Liste von Aufgaben, anhand der Aufgaben-Klasse)
Weitere Ziele:
-

6.1.3 Iteration 3

Aufwand:

Tabelle 10: Aufwand Iteration 3

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Use-Cases von 3.0 bis 3.2 werden ausformuliert. (fully-dressed/casual)	5	4	schrom01	Ja
2	Eine Liste von Pflanzen können im GUI abgebildet werden.	1	1	giavaphi	Ja
3	Einzelne Pflanzen können ausgewählt werden und genauere Information können abgebildet werden.	2	2	giavaphi	TW
4	Bilder der Pflanzen werden abgebildet, wenn diese ausgewählt worden sind.	3	2	giavaphi	TW
5	Die Pflanzenliste kann nach Saison, Klimazone und Suchanfrage gefiltert werden.	2	2	giavaphi	TW
6	Model für die Pflanzenliste wird erstellt und mit dem Controller verbunden.	7	7	schrom01	Ja
7	Testfälle für das Model der Pflanzenliste werden erstellt.	10	10	csomoeli	Ja
8	Adapter-Klasse für die Liste der geplanten Pflanzen (Crop) wird erstellt.	12	12	gulerdav	Ja
9	Adapter-Klasse für die Liste der Aufgaben wird erstellt.	11	11	huttegia	Ja
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	40	40	Alle Mitglieder	-
	Total	**100**	**95**		

Massnahmen:

Tabelle 11: Massnahmen Iteration 3

Massnahmen nicht/teilweise erfüllte Ziele:
Einzelne Pflanzen können ausgewählt werden und genauere Information können abgebildet werden: Die Daten der Pflanzen sind, im Moment noch, nur auf die Beschreibung begrenzt. Dies wird für den Prototyp als genügend gefunden und wird für den Beta-Release noch erweitert.
Bilder der Pflanzen werden abgebildet, wenn diese ausgewählt worden sind: Die Bilder werden nicht skaliert abgebildet. Dies wurde in der nächsten Iteration behoben.
Die Pflanzenliste kann nach Saison, Klimazone und Suchanfrage gefiltert werden: Filterung nach den Kriterien ist möglich, jedoch überschreiben sich die verschiedenen Filteroptionen. Für den Prototypen wurde dies als akzeptabel eingeschätzt, wird jedoch vor dem Beta-Release behoben werden.
Weitere Massnahmen (erfüllte Ziele):
-
Weitere Massnahmen:

Beim Laden der Pflanzenliste von der JSON-Datei, kommt es im Moment noch zu kleinen Verzögerungen. Bei grösseren Datensätzen könnte dies grösseren Zeitaufwänden führen. Daher wurde ein neuer Eintrag in die Risikoliste (siehe Tabelle 21) vorgenommen, dass die Leistung von der Pflanzenliste verbessert werden soll. Jedoch ist dies eine niedrige Priorität, da die Funktionalität des Produktes priorisiert wird.

Planung nächste Iteration:

Tabelle 12: Planung Iteration 4

Design:
Use-Cases 4.0 bis 5.2 werden ausformuliert. (brief/casual)
Alle Artefakte, welche eine Relevanz für den Prototypen haben, werden fertig erstellt.
Technischer Bericht 1 erstellt.
GUI:
Anpflanzzeit/Erntezeit kann im GUI ausgewählt werden und abgespeichert werden.
Detaillierte Ansicht der geplanten Pflanzen ist erhältlich. (inklusive Aufgabenliste)
Die Aufgabenliste kann bearbeitet und erweitert werden.
Im Zeitplan können die Aufgaben der nächsten sieben Tage abgerufen werden.
Implementation:
Model für die zeitlich geplanten Pflanzen wird erstellt und mit den Controllern verbunden.
Model für die Aufgabenliste wird erstellt und mit den Controllern verbunden.
Die Testfälle für den Model der zeitlich geplanten Pflanzen erstellt.
Die Testfälle für den Model der Aufgabenliste erstellt.
IO:
-
Weitere Ziele:
Implementation des Prototyps.
Prototyp erstellt, mit allen Grundfunktionalitäten: - Darstellung und Filterung von Pflanzen - Planung von Anpflanz- und Erntezeit - Aufgabenliste bearbeiten können - Aufgaben der folgenden Woche abfragen können

6.1.4 Iteration 4

Aufwand:

Tabelle 13: Aufwand Iteration 4

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Use-Cases 4.0 bis 5.2 werden ausformuliert. (brief/casual)	6	6	schrom01	Ja
2	Alle Artefakte, welche eine Relevanz für den Prototypen haben, werden fertig erstellt.	10	12	huttegia	Ja
3	Technischer Bericht 1 erstellt.	2	2	csomoeli	Ja
		2	4	giavaphi	
		2	4	gulerdav	
		2	4	huttegia	
		2	3	schrom01	
4	Anpflanzzeit/Erntezeit kann im GUI ausgewählt werden und abgespeichert werden.	2	5	giavaphi	Ja
5	Detaillierte Ansicht der geplanten Pflanzen ist erhältlich. (inklusive Aufgabenliste)	3	5	giavaphi	Ja
6	Die Aufgabenliste, kann bearbeitet und erweitert werden.	2	3	giavaphi	Nein
7	Im Zeitplan können die Aufgaben der nächsten sieben Tage abgerufen werden.	1	3	giavaphi	TW
8	Model für die zeitlich geplanten Pflanzen wird erstellt und mit den Controllern verbunden.	7	7	gulerdav	Ja
9	Model für die Aufgabenliste wird erstellt und mit den Controllern verbunden.	6	6	schrom01	Ja
10	Die Testfälle für den Model der zeitlich geplanten Pflanzen erstellt.	5	4	csomoeli	Ja
11	Die Testfälle für den Model der Aufgabenliste erstellt.	5	5	csomoeli	Ja
12	Implementation des Prototyps	4	4	gulerdav	Ja
13	Prototyp erstellt, mit allen Grundfunktionalitäten	-	-	-	Ja
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	40	40	Alle Mitglieder	-
	Total	**100**	**119**		

Massnahmen:

Tabelle 14: Massnahmen Iteration 4

Massnahmen nicht/teilweise erfüllte Ziele:
Die Aufgabenliste, kann bearbeitet und erweitert werden: Wegen fehlender Zeit, ist entschieden worden, diese in die nächste Iteration aufzunehmen.
Im Zeitplan können die Aufgaben der nächsten sieben Tage abgerufen werden: Es gibt im Moment keine Testwerte, welche in der Benutzeroberfläche abgerufen werden können. Deshalb wird der nächsten Iteration die Werte erweitert, dass eine bessere Demonstration möglich ist.
Weitere Massnahmen (erfüllte Ziele):
-
Weitere Massnahmen:
Während der Erstellung des technischen Berichtes und des Prototyps, wurde realisiert, dass mehrere Designs und Artefakte optimiert werden können. Es wurde entschieden, dass das Projekt in der nächsten Iteration leicht geändert wird und die Artefakte und Designs erneut überarbeitet werden.
Namensgebung von Klassen und Methoden werden in der nächsten Iteration überarbeitet werden.

Planung nächste Iteration:

Tabelle 15: Planung Iteration 5

Design:
Artefakte und Design werden überarbeitet.
GUI:
Stylesheet für Design des GUI wird eingefügt werden.
Benachrichtigungen über Änderungen durch Wetterdienst.
Implementation:
Namensgebung der Klassen und Methoden wird überarbeitet.
Standort der geplanten Pflanzen können definiert werden.
Wetterdienst API wird eingefügt werden.
Testfälle für Wetterdienst werden erstellt.
Senden von Benachrichtigungen von Änderungen.
IO:
-
Weitere Ziele:
-

6.1.5 Iteration 5

Aufwand:

Tabelle 16: Aufwand Iteration 5

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Artefakte und Design werden überarbeitet	5	5	huttegia	Ja
2	Klassendiagramm wird für Wetterdienst überarbeitet	4	3	huttegia	Ja
3	Stylesheet für Design des GUI wird eingefügt werden	2	2	giavaphi	Ja
4	GUI-Design wird überarbeitet	8	8	giavaphi	TW
5	Benachrichtigungen über Änderungen durch Wetterdienst	3	4	schrom01	TW
6	Wachstumsphasengruppen können bestimmt werden.	3	3	schrom01	Ja
7	Namensgebung der Klassen und Methoden wird überarbeitet	4	4	gulerdav	Ja
8	Standort der geplanten Pflanzen können definiert werden	3	3	giavaphi	Ja
9	Wetterdienst API wird eingefügt werden	3	3	huttergia	TW
10	Testfälle für Wetterdienst werden erstellt	6	5	csomoeli	Ja
11	Testfälle für die Pflanzenobjekte werden erstellt	4	4	csomoeli	Ja
12	Senden von Benachrichtigungen von Änderungen	9	9	schrom01	Ja
13	Erstellen eines Tutorials für das Projekt	5	5	gulerdav	TW
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	40	40	Alle Mitglieder	-
	Total	**101**	100		

Massnahmen:

Tabelle 17: Massnahmen Iteration 5

Massnahmen nicht/teilweise erfüllte Ziele:
GUI-Design wird überarbeitet: Noch nicht das ganze Design konnte überarbeitet werden und wird zusammen mit dem Style Sheet in der nächsten Iteration überarbeitet.
Benachrichtigungen über Änderungen durch Wetterdienst: Die Benachrichtigungen vom Wetterdienst können noch nicht getestet werden, da der Wetterdienst noch nicht komplett implementiert worden ist.
Wetterdienst API wird eingefügt werden: Die Klassen, um die API-Anfragen zu bearbeiten wurden schon erstellt, jedoch konnte aufgrund von Problemen mit der Lizenz der API-Anbieter, noch keine eingebunden werden.
Erstellen eines Tutorials für das Projekt: Noch fehlende Screenshots, welche nach der Überarbeitung des GUI noch eingefügt werden.

Planung nächste Iteration:

Tabelle 18: Planung Iteration 6

Design:
Technischer Bericht 2 wurde erstellt
Javadoc Dokumentation wurde in allen Klassen überprüft und nachgefasst
GUI:
Style wird für den Beta-Release vervollständigt.
Implementation:
Aufgabenliste wird überarbeitet. (für eine bessere Darstellung und Benutzerfreundlichkeit)
Restliche Testfälle werden geschrieben und kontrolliert
Benachrichtigungssystem komplett integriert
Wetterdienst wird komplett integriert
Tutorial wird vervollständigt
IO:
-
Weitere Ziele:
Beta-Version wird erstellt.
Kontrolle Clean-Code Richtlinien eingehalten.
Benutzer Testfälle werden getestet im GUI (mit gegebenen Testdaten)

6.1.6 Iteration 6

Aufwand:

Tabelle 19: Aufwand Iteration 6

	Ziel	Geplant (h)	Ist (h)	Verantwortlich	Erfüllt
1	Technischer Bericht 2 erstellt.	5	4	csomoeli	Ja
		5	5	giavaphi	
		5	6	gulerdav	
		5	4	huttegia	
		5	5	schrom01	
2	Javadoc Dokumentation wurde in allen Klassen überprüft und nachgefasst	5	4	csomoeli	Ja
3	Style wird für den Beta-Release vervollständigt	5	6	giavaphi	Ja
4	Aufgabenliste wird überarbeitet	6	5	schrom01	Ja
5	Restliche Testfälle werden geschrieben und kontrolliert	5	5	csomoeli	Ja
6	Benachrichtigungssystem komplett integriert	4	5	schrom01	Ja
7	Wetterdienst wird komplett integriert	4	4	huttergia	TW
8	Tutorial wird vervollständigt	3	3	gulerdav	Ja
9	Beta-Version wird erstellt	3	4	huttergia	Ja
10	Kontrolle Clean-Code Richtlinien eingehalten	5	5	gulerdav	Ja
11	Benutzer Testfälle werden getestet (GUI)	3	3	giavaphi	Ja
	SWEN1: Übungsaufgaben	2	2	Alle Mitglieder	-
	Sitzungen (2x wöchentlich)	40	40	Alle Mitglieder	-
	Total	**110**	**111**		

Massnahmen:

Tabelle 20: Massnahmen Iteration 6

Massnahmen nicht/teilweise erfüllte Ziele:
Wetterdienst wird komplett integriert: Wegen der fehlenden API haben wir uns entschieden, bis eine Lizenz für die Wetter API erworben ist das Wetter zu simulieren für die Demostation des Produktes.
Weitere Massnahmen (erfüllte Ziele):
-
Weitere Massnahmen:
Aus Zeitlichen Gründen wurde entschieden, dass die Implementation von Pestiziden weggelassen werden soll.

6.1.7 Risikotabellen der Meilensteine

Die Risikotabelle (siehe Tabelle 21) wurde in der Projektskizze erstellt für den Meilenstein 1.

Tabelle 21: Risikotabelle Meilenstein 1

#	Name	Beschreibung	Wahr-schein-lichkeit	Scha-dens-poten-tial	Prio-rität	Massnahmen
1	Datenbe-schaffung (Pflanzen)	Abfrage von Pflanzen, welche die benötigten Informationen erhal-ten für die Applikation	Hoch	Hoch	1	Recherche im Internet und Anfrage an Gärtner (Experten). Für eigene Datenbank.
2	Fehlendes Wissen Frontend	Die Teammitglieder haben wenig Erfah-rung in Frontend	Hoch	Hoch	2	Bis Meilenstein 2 soll mindestens ein Teammit-glied sich mit JavaFX aus-einandersetzten und den Prototypen entwickeln.
3	Wetterein-bindung 1	Die Schnittstelle des externen Wetterdiens-tes, könnte sich je nach Standort ändern	Mittel	Tief	3	Extraklasse für verschie-dene Schnittstellen er-stellen, welche von ei-nem Interface vorgege-ben werden.
4	Wetterein-bindung 2	Ausfall von des Wet-terdienstes API. (Ab-frage nach Wetter nicht möglich)	Tief	Tief	4	Ersatz-Wettereinbindung auswählen, für, falls die erste Ausfällt.

Die Risikoliste für Meilenstein 2 (siehe Tabelle 22) hat nur kleine Änderungen vollzogen:

Die Risiken Nummer 1 und 2 wurden nicht behoben, bestenfalls reduziert. Indem ein Experte in Form eines lokalen Bauern gefunden worden ist und uns mit einer Sample-Daten zur Verfügung gestellt hat. Weitere Daten können in kleinen wöchentlichen Updates erweitert werden. Das Wissen des Fronend-Teams hat sich in den letzten Wochen verbessert, jedoch fehlen immer noch gewisse Grundsätze, welche bis zu der nächsten Iteration ausgefeilt werden, damit ein akzeptables Design erstellt werden kann.

Die restlichen Risiken von Meilenstein 1 wurden nicht behandelt, da diese Implementation erst in Meilenstein 3 geplant worden ist.

Wegen schlechter Performance der Darstellung der Pflanzenliste wurde diese als Risiko aufgenommen. Weil bisher nur wenig Daten für den Beta-Release geplant worden sind, wurde dieses Risiko niedrig eingestuft. Massnahmen dagegen wurde auf eine Proxy-Klasse reduziert, welche die Daten erst abruft, wenn diese benötigt werden und nur Änderungen berechnet werden müssen.

Tabelle 22: Risikotabelle Meilenstein 2

#	Name	Beschreibung	Wahrscheinlichkeit	Schadenspotential	Priorität	Massnahmen
1	Datenbeschaffung (Pflanzen)	Abfrage von Pflanzen, welche die benötigten Informationen erhalten für die Applikation	Mittel	Tief	4	Recherche im Internet und Anfrage an Gärtner (Experten). Für eigene Datenbank.
2	Fehlendes Wissen Frontend	Die Teammitglieder haben limitierte Erfahrung in Frontend	Mittel	Tief	1	Bis Meilenstein 3 Wissen vertiefen und genügendes Style für Beta-Release entwickeln
3	Wettereinbindung 1	Die Schnittstelle des externen Wetterdienstes, könnte sich je nach Standort ändern	Mittel	Tief	2	Extraklasse für verschiedene Schnittstellen erstellen, welche von einem Interface vorgegeben werden.
4	Wettereinbindung 2	Ausfall von des Wetterdienstes API. (Abfrage nach Wetter nicht möglich)	Tief	Tief	5	Ersatz-Wettereinbindung auswählen, für, falls die erste Ausfällt.
5	Performance	Beim Laden der Pflanzenliste gibt es eine kleine Pause von einer halben Sekunde bis zwei Sekunden, bevor die Daten dargestellt werden. (Nur drei Einträge)	Mittel	Tief	3	Erstellung einer Proxy Klasse für die Liste der Pflanzen.

Die Risiken haben sich seit Meilenstein 2 nicht gross geändert, ausser dass das bisher mangelhafte Wissen des Frontend-Teams erweitert worden ist, dass dieses in der Zukunft nur noch kleine Herausforderungen mit sich bringt. Da dies kein genügendes Risiko mehr darstellt, wurde es in der Risikotabelle von Meilenstein 3 entfernt (siehe Tabelle 23).

Tabelle 23: Risikotabelle Meilenstein 3

#	Name	Beschreibung	Wahrscheinlichkeit	Schadenspotential	Priorität	Massnahmen
1	Datenbeschaffung (Pflanzen)	Abfrage von Pflanzen, welche die benötigten Informationen erhalten für die Applikation	Mittel	Tief	3	Recherche im Internet und Anfrage an Gärtner (Experten). Für eigene Datenbank.
2	Wettereinbindung 1	Die Schnittstelle des externen Wetterdienstes, könnte sich je nach Standort ändern	Mittel	Tief	1	Extraklasse für verschiedene Schnittstellen erstellen, welche von einem Interface vorgegeben werden.
3	Wettereinbindung 2	Ausfall von des Wetterdienstes API. (Abfrage nach Wetter nicht möglich)	Tief	Tief	4	Ersatz-Wettereinbindung auswählen, für, falls die erste Ausfällt.
4	Performance	Beim Laden der Pflanzenliste gibt es eine kleine Pause von einer halben Sekunde bis zwei Sekunden, bevor die Daten dargestellt werden. (Nur drei Einträge)	Mittel	Tief	2	Erstellung einer Proxy Klasse für die Liste der Pflanzen.

6.2 Testresultate

Testabdeckung

Die Abdeckung der Nicht-GUI-Klassen durch die automatisierten Tests (Modul- und Integrationstests) stellt sich wie folgt dar:

Tabelle 24: Testresultate/Abdeckung

Package (ch.zhaw...)	**Klassen**	**Methoden**	**Zeilen**
gartenverwaltung.types	100% (20/20)	91% (112/122)	84% (294/346)
gartenverwaltung.models	100% (8/8)	77% (68/88)	85% (210/246)
gartenverwaltung.json	100% (6/6)	100% (14/14)	84% (42/50)
gartenverwaltung.io	100% (12/12)	94% (68/72)	94% (316/336)

Created with IntelliJ

6.3 Installations- und Gebrauchsanweisung

6.3.1 Vorbedingungen

- Das Java Runtime Environment (JRE) muss in der Version 17 auf Ihrem Computer installiert sein.
- Das Java Development Kit (JDK) muss in der Version 17 auf Ihrem Computer installiert sein.

6.3.2 Installation und Starten der Applikation

1. Laden sie die neuste Version der Applikation herunter. Sie finden die Applikation hier: https://github.zhaw.ch/schrom01/PM3-HS22-IT21b_WIN-Team1/tags
2. Extrahieren sie die heruntergeladene ZIP-Datei
3. Öffnen sie ein Kommandozeilenfenster und navigieren sie an den Speicherort der Entpackten Applikation.
4. Führen sie den Befehl: «./gradlew.bat run» (auf Windows) oder «./gradlew run» (auf macOS und Linux) aus.

6.3.3 Anwendung

6.3.3.1 Übersicht und allgemeine Informationen

Um eine Übersicht über die in der Applikation verfügbaren Funktionen und allgemeine Informationen zu erhalten, starten sie die Anwendung und klicken sie auf das folgende Symbol oben links im Menu: 

6.3.3.2 Basiseinstellungen

Um die Basiseinstellungen zu ändern, klicken sie auf das folgende Symbol oben rechts im Menu: 

Aktivieren sie **„Show Tutorial Menu“**, falls sie noch wenig Erfahrung mit der Anwendung haben. Das Tutorial dient als Nachschlagewerk und erscheint nach dem Aktivieren erscheint als zusätzliche Schaltfläche im Menu die angeklickt werden kann, um das Tutorial zu öffnen.

Stellen sie bei **„Select Default Hardiness Zone“** die Klimazone ein, in welcher sich Ihr Garten befindet. Diese Einstellung dient dazu, dass Ihnen nur Pflanzen angezeigt werden, die in Ihrer Klimazone angepflanzt werden können, und dass die Pflege Ihrer Pflanzen von der Applikation entsprechend angepasst wird.

Mit einem Klick auf «Set Location» kann der Ort eingetragen werden, an welchem sich ihr Garten befindet. Der Ort wird benötigt damit die Applikation die richtigen Wetterdaten verwenden kann.

6.3.3.3 Pflanzenübersicht

Um eine Übersicht über die in Ihrem Garten vorhandenen oder geplanten Pflanzen zu erhalten, klicken sie auf «My Garden» oben im Menu.

Um eine Pflanze zu **löschen,** klicken sie auf das folgende zugehörige Symbol: 

Um mehr Informationen zu einer Pflanze zu erhalten, klicken sie auf das folgende Symbol:
Es öffnet sich ein neues Fenster mit allen Informationen zur ausgewählten Pflanze sowie auch die zugehörigen geplanten Aufgaben.
Die Aufgaben können ebenfalls mit einem Klick auf das Symbol  gelöscht werden.
Um eine Aufgabe zu bearbeiten, klicken sie auf das Symbol  .
Um neue Aufgaben hinzuzufügen, klicken sie auf das Symbol
Mit einem Klick auf «Set Area» kann die Fläche eingestellt werden.
Um das Fenster wieder zu schliessen, klicken sie auf «Go Back».

6.3.3.4 Pflanzen hinzufügen

Um eine neue Pflanze im Garten hinzuzufügen, öffnen sie zuerst die Pflanzenübersicht, indem sie im Menu auf «My Garden» klicken.

Mit dem Button «Add new Plant» unterhalb der Liste können sie eine neue Pflanze hinzufügen.

Sie können direkt eine Pflanze aus der Liste auswählen oder mit dem Textfeld oberhalb der Liste und mit den Optionen auf der rechten Seite die Liste filtern. Wenn sie die ID der gewünschten Pflanze kennen, können sie diese mit «#» eintippen. Geben sie beispielsweise «#1» im Textfeld ein, wenn sie die Pflanze mit der ID «1» suchen.

Sobald sie die richtige Pflanze ausgewählt haben, klicken sie auf «Select Harvest/Sow Day».

Es öffnet sich ein neues Fenster. Wählen sie dort aus, ob sie das Anpflanz- oder Erntedatum eingeben wollen. Klicken sie anschliessend auf das Kalender Symbol neben dem Textfeld, um das entsprechende Datum auszuwählen.

Bestätigen sie mit «Save» das ausgewählte Datum.

6.3.3.5 Aufgabenübersicht

Um eine Übersicht der in den nächsten sieben Tage fälligen Aufgaben zu erhalten, klicken sie auf «My Schedule» oben im Menu.

Auf der rechten Seite finden sie eine Auflistung der Aufgaben, die in den nächsten Tagen fällig werden. Klicken sie auf «Task completed», wenn sie eine Aufgabe abgeschlossen haben, damit diese als erledigt markiert wird und aus dieser Ansicht entfernt wird.

Auf der linken Seite können sie eine bestimmte Pflanze anklicken, um nur die Aufgaben anzuzeigen, die mit der ausgewählten Pflanze verknüpft sind.

6.4 Verzeichnisse

6.4.1 Glossar

A

API	Einige Schnittpunkte, um Funktionen von einem externen Programm zu verwenden

B

Bibliothek	*Sammlung von bereits geschriebenen Code-Teilen*

C

cachen	*Zwischenspeichern um mehrfaches laden zu vermeiden.*
Code-Duplizierung	*Mehrfaches Vorkommen desselben Codes*

D

Dependency-Injection	*Einfügen von Abhängigkeiten*

F

Filtern	*Bedingtes Anzeigen*
Framework	*Vordefinierte Struktur, die die Entwicklung vereinfacht*

G

GUI	*Grafische Benutzeroberfläche*

I

IDE	*Itegrierte Endwicklungsumgebung*
instanziiert	*Erzeugen eines Objektes anhand einer Klasse*
IO	*Input und Output von Dateien*

J

Java	*Programmiersprache*
JSON	*JavaScript Object Notation (Textformat zur Beschreibung hierarchischer Daten)*

K

Klassen	*Gruppierung von Daten und zugehöriger Funktionalität*

L

Logging	*Fehlerprotokollierung*

P

Pattern	*Konzeptionelle Struktur (von Code)*
Persistenz	*Daten unabhängig des laufenden Systems speichern.*
Plant	*Eine Objektklasse, die eine Pflanze repräsentiert, die alle Infromationen zu einer Pfalnzenart hält.*

R

Reflection	*Abfrage der rsprünglichen Klassenstruktur zur Laufzeit des Programms*

T

TKP	*Einnahmen pro 1000 Aufrufe von Werbung*

X

XML	*Extensible Markup Language (Textformat zur Beschreibung hierarchischer Daten)*

6.4.2 Literaturverzeichnis

[1] Growing Interactive GmbH, „GrowVeg," Growing Interactive GmbH, o.D. [Online]. Available: https://www.growveg.co.uk/garden-planner-intro.aspx. [Zugriff am 24 September 2022].

[2] o.N, „Vegplotter," o.D. [Online]. Available: https://vegplotter.com/. [Zugriff am 24 September 2022].

[3] Green Living Solution AG, „Smartgardener," Green Living Solution AG, o.D. [Online]. Available: https://www.smartgardener.com. [Zugriff am 24 September 2022].

[4] o.N, „VeggieGardenPlaner," Bentosoftware, o.D. [Online]. Available: https://play.google.com/store/apps/details?id=com.bentosoftware.gartenplaner&hl=en_US&gl=US. [Zugriff am 24 September 2022].

[5] wetter.com GmbH, „Meteonomiqs," wetter.com GmbH, o.D. [Online]. Available: https://www.meteonomiqs.com/de/wetter-api/?utm_source=google&utm_medium=api&utm_campaign=de&gclid=Cj0KCQiAvqGcBhCJARIsAFQ5ke4lrWHrVvADiaaWPUdgX-mBs6D8YbTbfgQ2_y6DjuvLPHOsp03_n9YaAtW4EALw_wcB. [Zugriff am 1 12 2022].

6.4.3 Abbildungsverzeichnis

Abbildungen:

Tabellen: